Gestaltung der Mensch-Computer-Interaktion zur Entscheidungsunterstützung in Planungssystemen

Von der Fakultät für MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik
der Brandenburgischen Technischen Universität Cottbus-Senftenberg

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Master of Science

Anna Prenzel

geboren am 09.08.1987 in Görlitz

Gutachter: Prof. Dr. rer. nat. habil. Petra Hofstedt

Guatchter: Prof. Dr.-Ing. habil. Rainer Groh

Gutachter: Prof. Dr.-Ing. Georg Ringwelski

Gutachter: Prof. Dr.-Ing. Uwe Meinberg

Tag der mündlichen Prüfung: 4.6.2018

Zusammenfassung

Mit Hilfe von modernen Feinplanungssystemen kann die Produktions-, Flotten- oder Personaleinsatzplanung nahezu vollständig automatisiert werden. Für den Disponenten wird dadurch der Planungsaufwand erheblich reduziert. Gleichzeitig bieten diese Systeme dem Disponenten jedoch häufig nur unzureichende Interaktionsmöglichkeiten, um sein eigenes Expertenwissen in die Planerstellung einfließen zu lassen. Das Finden und Umsetzen von passenden Planungsentscheidungen ist häufig ein aufwändiger und fehleranfälliger Prozess, der vom Computer nicht unterstützt wird. Dieser Mangel kann die Brauchbarkeit der Pläne und damit den Nutzen des Planungssystems erheblich beeinträchtigen.

In dieser Arbeit wird ein Konzept für die Gestaltung der Interaktion zwischen Mensch und Computer in Feinplanungssystemen entwickelt, mit dem eine effektive Einbeziehung des Disponenten abgesichert werden kann. Im Gegensatz zu herkömmlichen Interaktionskonzepten baut es auf wissenschaftlichen Erkenntnissen über menschliche Entscheidungs- und Entwurfsprozesse auf. Dadurch ist es möglich, eine Computerunterstützung sowohl für die Ermittlung als auch für die Umsetzung von Planungsentscheidungen durch den Disponenten zu definieren. Zudem ist das Konzept auf beliebige Feinplanungsprobleme anwendbar. Es setzt sich aus 9 Interaktionsrichtlinien zusammen, deren Effektivität in einem Anwendertest nachgewiesen wird.

Abstract

With the help of modern detailed planning systems, production, fleet or personal resource planning can be automated nearly completely. By this, the planning effort for the scheduler is reduced significantly. However, these systems often provide insufficient interaction possibilities for the scheduler to incorporate his own expert knowledge in the creation of the schedule. Finding and implementing suitable planning decisions is often a time-consuming and error-prone process that is not supported by the computer. This deficiency can impair the practicability of the schedules and hence the benefit of the planning system considerably.
In this work, a concept for the design of the human-computer-interaction in detailed planning systems is developed. It helps to ensure an effective incorporation of the human scheduler. Unlike conventional interaction concepts the concept is built on scientific research about human decision-making and design processes. This makes it possible to define computer support for both the formation and the implementation of planning decisions by the scheduler. Furthermore the concept can be applied to any detailed scheduling problem. It is composed of 9 interaction guidelines. Their effectiveness is proven by a user test.

Danksagung

Ich freue mich sehr, mit diesem Text das Ergebnis meiner Forschungsarbeit vorlegen zu dürfen. An erster Stelle möchte ich mich bei den Personen bedanken, die mich bei der Anfertigung dieser Arbeit unterstützt haben. Der größte Dank geht an meine Eltern, die mir den Rücken frei gehalten und mich in jeder Hinsicht gestärkt haben. Ein ganz besonderer Dank geht an meine Betreuer Professor Petra Hofstedt und Professor Georg Ringwelski, die mir diese Arbeit ermöglicht haben und mir fortwährend mit Rat und Tat zur Seite standen. Außerdem möchte ich allen Personen danken, die an der Durchführung der Fallstudien beteiligt waren: Jens Heider, Daniel Tasche, Romy Müller, Steve Dillan und Maria Isendahl waren an der Entwicklung des Prototyps für die Flottenplanung beteiligt und haben zum Gelingen des Anwendertests beigetragen. Benny Höckner war stets ein hilfreicher Ansprechpartner für Fragen rund um die Stundenplanung am Lehrstuhl für Programmiersprachen und Compilerbau der Brandenburgischen Technischen Universität Cottbus-Senftenberg.

Inhaltsverzeichnis

1 Einführung

In diesem Kapitel erfolgt die Einführung in die Thematik der interaktiven Planung und die Herleitung der Forschungsfrage. Zunächst wird die Planung aus betriebswirtschaftlicher Sicht betrachtet (Abschnitt 1.1). Dabei werden unterschiedliche Arten von Planungsentscheidungen und Planungsphasen sowie die Rolle des Disponenten und der Aufbau von Planungssystemen erläutert. Danach werden Kriterien der Nutzerakzeptanz eingeführt, die für den erfolgreichen Einsatz von Planungssystemen entscheidend sind. In Abschnitt 1.3 wird der aktuelle Stand der Forschung zur Verbesserung der Nutzerakzeptanz untersucht. Daraus wird in Abschnitt 1.4 die Forschungsfrage abgeleitet, die in dieser Arbeit behandelt wird.

1.1 Einführung in die Planung

In diesem Abschnitt werden die wichtigsten Aspekte der Planung aus betriebswirtschaftlicher Sicht vorgestellt.

1.1.1 Planung als Schlüsselfaktor für wirtschaftlichen Erfolg

Das Ziel eines Unternehmens ist in der Regel die Erwirtschaftung von Gewinn oder der Ausgleich von Verlusten (bei Non-Profit-Organisationen). Der Erfolg und die Wettbewerbsfähigkeit eines Unternehmens werden von der Zufriedenheit seiner Kunden bestimmt. Zufriedenheit kann durch eine hohe Qualität der angebotenen Produkte und Dienstleistungen erreicht werden. Ein wesentlicher Faktor ist aber auch die Pünktlichkeit und Schnelligkeit, mit der die Leistungen bereit gestellt werden. Beispielsweise sind beim Versand von Produkten kurze Lieferzeiten erwünscht. Die logistische Leistungsfähigkeit eines Unternehmens hängt von seiner internen Organisation ab: Ein Unternehmen besitzt verschiedenartige Ressourcen,

Abbildung 1.1: Planung zur Koordination von Aufträgen und Ressourcen unter Berücksichtigung von Zielen

wie z.B. qualifizierte Mitarbeiter und spezialisierte Fahrzeuge und Maschinen. Das Zusammenspiel dieser wertschöpfenden Elemente muss durch eine übergeordnete Instanz (Unternehmensleitung, Management) koordiniert werden. Diese ist dafür verantwortlich, dass das Unternehmen bei einer ausgewogenen Beanspruchung ihrer Ressourcen alle Leistungen zur Zufriedenheit der Kunden erfüllt. Gleichzeitig muss sie sicherstellen, dass langfristig die Unternehmensziele erreicht werden.
Die Koordination besteht im Wesentlichen aus der Zuordnung von Arbeitsaufgaben zu Ressourcen und wird als *Planung* oder *Disposition* bezeichnet. Abbildung 1.1 zeigt die Zusammenhänge, die bei der Planung berücksichtigt werden müssen. Im Mittelpunkt stehen die Organisation und ihre Kunden. Die Kunden beantragen die Inanspruchnahme von Diensten der Organisation, indem sie Aufträge erstellen. Die Aufträge müssen den Ressourcen zugeteilt werden, wobei die Bedingungen zu beachten sind, unter denen die Ressourcen arbeiten. Dazu zählt die Bereitstellung benötigter Materialien, die Einhaltung von Arbeitszeiten der Mitarbeiter oder die Berücksichtigung von Maschinen- und Fahrzeugkapazitäten. An die Aufträge sind ebenfalls Bedingungen geknüpft, wie z.B. Terminanforderungen oder Qualitätsansprüche. Die Bedingungen von Aufträgen und Ressourcen schränken den Handlungsspielraum bei der Planung ein.
Unternehmen unterschiedlicher Branchen haben oft ähnliche *Planungsprobleme*. Während ein Logistikdienstleister Transportmittel als Ressourcen einsetzt, benö-

tigt ein Produktionsunternehmen Maschinen zur Fertigung von Teilen. In beiden Fällen müssen die anstehenden Aufträge termingerecht verplant werden: Das Logistikunternehmen legt Fahrzeuge zur Auslieferung von Gütern fest, der Fließbandproduzent koordiniert die Bearbeitung von Teilen auf seinen Maschinen. Sowohl bei der Produktionsplanung als auch bei der Tourenplanung gilt es, die Durchlaufzeiten von Aufträgen zu minimieren und die vorhandenen Ressourcen optimal auszulasten. Dieses Prinzip kann auch auf Personal- und Schichtplanungsprobleme übertragen werden. Hier entspricht ein Arbeitsauftrag der Erfüllung einer Schicht. In einem Krankenhaus muss z.B. in der Frühschicht eine ausreichende Anzahl an Schwestern anwesend sein, wobei Überstunden nach Möglichkeit vermieden werden sollen.
Um die Einhaltung der Bedingungen und die Maximierung des Gewinns zu gewährleisten, werden im Rahmen der Disposition *Pläne* erstellt, auf deren Grundlage die Ressourcen eingesetzt werden (vgl. Abbildung 1.1). Aus ihnen kann abgelesen werden, zu welchem Zeitpunkt und von welcher Ressource ein bestimmter Auftrag bearbeitet wird. Je nach Anwendungsgebiet versteht man darunter z.B. Tourenpläne, Schichtpläne oder Produktionspläne.

1.1.2 Arten von Planungsentscheidungen

Es können operative, taktische und strategische Planungsentscheidungen unterschieden werden.
Operative Planungsentscheidungen beschäftigen sich mit der Umsetzung konkreter oder prognostizierter Aufträge durch eine optimale Ausnutzung der vorhandenen betrieblichen Ressourcen. Die Rahmenbedingungen für operative Entscheidungen werden im Großen und Ganzen durch die Struktur der Wertschöpfungskette vorgegeben, durch die z.B. Produktions- oder Transportprozesse festgelegt werden. Im Einzelnen wird der Handlungsspielraum für operative Entscheidungen u.a. eingeschränkt durch die Produktionsraten von Maschinen, die Transportkapazität von Fahrzeugen und die Verfügbarkeit von Personal (Günther und Tempelmeier, 2012).
Taktische Planungsentscheidungen werden getroffen, um die betriebliche Infrastruktur an die zukünftigen Anforderungen anzupassen (Kasprik, 2002; Günther und Tempelmeier, 2012). Sie stellen sicher, dass die Organisation mittelfristig alle Voraussetzungen erfüllt, um die operativen Prozesse durchführen zu können. Ein Produktionsunternehmen könnte beispielsweise die Optimierung der Arbeitsabläu-

fe seiner Fließproduktionssysteme oder eine Anpassung der Lagerstrategie vornehmen (Fabrikplanung oder Infrastrukturplanung, vgl. Günther und Tempelmeier (2012)). Bei einem Logistikunternehmen regeln taktische Entscheidungen z.B. die Neubeschaffung oder Reduzierung von Fahrzeugen. Insgesamt werden durch taktische Entscheidungen die Rahmenbedingungen und der Handlungsspielraum für operative Entscheidungen definiert.
Gegenstand der langfristigen Planung, die oft mehrere Jahre im Voraus erfolgt, ist die Klärung *strategischer* Fragen entlang der Lieferkette (Kasprik, 2002). Damit das Unternehmen auf Dauer wettbewerbsfähig bleibt und Gewinne erzielt, müssen Entscheidungen in Bezug auf die Auswahl von Lieferanten, Standorten und Produktsortimenten getroffen werden (Tompkins, 2010).

Beispiel 1.1 (strategische Entscheidungen): Ein Paketzusteller kann langfristig anhand von Kunden- und Auftragsdaten abschätzen, ob es sich lohnt, ein lokales Depot in einer bestimmten Region einzurichten.

1.1.3 Phasen der Planung

Im Folgenden wird die typische Vorgehensweise von Unternehmen und Organisationen bei der Planung erläutert. Sie besteht in einer hierarchischen Planung, die sich in die Phasen der Grob- und Feinplanung unterteilt. Beide Phasen dienen zur Festlegung operativer Planungsentscheidungen, mit denen die Ausführung konkreter oder prognostizierter Aufträge gesteuert wird.

Grobplanung

Planungsentscheidungen erstrecken sich über die gesamte Lieferkette (engl. supply chain) eines Unternehmens vom „Lieferanten des Lieferanten“ bis zum „Kunden des Kunden“ (Günther und Tempelmeier, 2012). So müssen in einem Produktionsunternehmen die Ausgangsmaterialien erst bei den Lieferanten bestellt und an die Produktionsorte transportiert werden, bevor die Herstellung der Produkte und die anschließende Verteilung in den Absatzgebieten erfolgen kann. Auch das Unternehmen selbst kann mehrere Standorte oder Werkstätten besitzen, zwischen denen ein Austausch von Gütern notwendig ist. Alle Elemente der Lieferkette müssen lang-

fristig (in der Regel mehrere Monate im Voraus) aufeinander abgestimmt werden. In der Regel ist es zu aufwändig, detaillierte Planungsentscheidungen für jede einzelne Ressource jedes Standorts im Voraus für die gesamte Lieferkette festzulegen. Stattdessen wird häufig eine Grobplanung vorgenommen. Dabei werden mehrere Ressourcen zu Ressourcengruppen und Aufträge zu Auftragsgruppen zusammengefasst (Staedler u. a., 2012). Mit diesen aggregierten Einheiten wird ein langfristiger, zeitlich grob strukturierter Plan erstellt, der den prognostizierten Bedarf an Gütern für die gesamte Lieferkette abdeckt. Die Auftragsgruppen werden so auf Standorte und Ressourcengruppen vorverteilt, dass die Transportwege entlang der Lieferkette möglichst kurz und die Bearbeitungskosten an den Standorten möglichst gering sind (Tompkins, 2010; Huber, 2011).
Die Erstellung von Grobplänen erfordert eine globale Sicht auf die Lieferkette. Von den Details der konkreten Auftragsausführung muss daher zunächst abstrahiert werden. Es wird z.B. noch nicht die konkrete Bearbeitungsreihenfolge einzelner Aufträge festgelegt. Außerdem werden bei der Grobplanung bestimmte auftrags- und ressourcenspezifische Bedingungen vernachlässigt. Dazu gehören z.B. Belade- und Entladezeiten von Transportmitteln oder Umrüstzeiten von Maschinen (Staedler u. a., 2012).

Feinplanung

Die *Feinplanung* ist ein Prozess, bei dem einzelne, nicht weiter zerlegbare Aufgaben konkreten Bearbeitungszeiten und Ressourcen zugewiesen werden. Dabei werden die Bedingungen von Aufgaben und Ressourcen vollständig berücksichtigt (Staedler u. a., 2012). Es entsteht ein Plan, der die Reihenfolge der Aufgabenbearbeitung für jeden Tag des Planungszeitraumes vorschreibt und den Mitarbeitern des Unternehmens oder der Organisation kurzfristig (in der Regel wenige Wochen im Voraus) ausgehändigt wird. Ist das Unternehmen in eine Lieferkette eingebunden, so ist der Feinplan die lokale Spezifizierung des Grobplans für einen einzelnen Unternehmensstandort bzw. für eine einzelne Abteilung.
Eine besondere Rolle spielt bei der Feinplanung die Festlegung der Reihenfolge, in der Aufgaben durchgeführt werden. Sie entscheidet über die Gesamtdauer, die eine Ressource zur Durchführung aller Aufgaben benötigt und beeinflusst somit die Kosten, die ihr Einsatz verursacht. Die Gesamtdauer setzt sich aus der Ausführungsdauer jeder Aufgabe und den benötigten Übergangszeiten zwischen jeweils

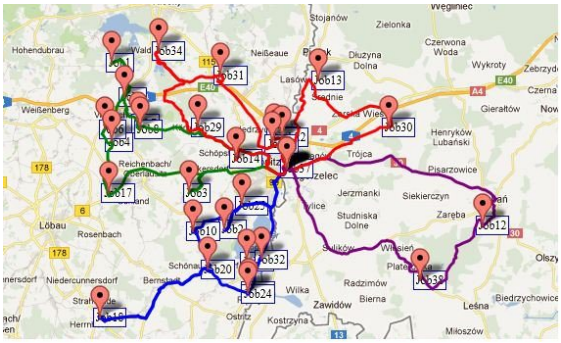


Abbildung 1.2: Tourenplan eines Logistikunternehmens, bei dem 4 Fahrzeuge eingesetzt werden.

zwei Aufgaben zusammen (Staedler u. a., 2012).

Beispiel 1.2: In einem Logistikunternehmen finden Aufträge an unterschiedlichen geographischen Orten statt (siehe Abbildung 1.2), sodass die Fahrzeiten zwischen den Orten berücksichtigt werden müssen. Während der Feinplanung wird eine Tour mit minimaler Länge gesucht (Problem des Handlungsreisenden, (Görz, Rollinger und Schneeberger, 2003)).

Beispiel 1.3: In einem Produktionsunternehmen gibt es verschiedene Typen von Produktionsaufträgen. Beim Wechsel des Produktionstyps auf derselben Maschine wird eine bestimmte Vorbereitungszeit zur Umrüstung benötigt. Die Durchlaufzeit der Aufträge ist kürzer, wenn Aufträge gleichen Typs am Stück ausgeführt werden.

Voraussagen über Fertigstellungstermine, die bereits anhand des Grobplanes getroffen wurden, müssen bei der Feinplanung unter Umständen leicht korrigiert werden. Außerdem kann es passieren, dass eine Umsetzung der Vorgaben im Grobplan aufgrund der Bedingungen, wie z.B. Kapazitätsbegrenzungen von Ressourcen, nicht möglich ist (Staedler u. a., 2012).

1.1.4 Planungssysteme

Im dieser Arbeit soll die Computerunterstützung für operative Planungsentscheidungen im Vordergrund stehen. Sowohl die Grob- als auch die Feinplanung kann

mit Hilfe von Softwaresystemen, die auch als APS-Systeme (APS = **A**dvanced **P**lanning and **S**cheduling) bezeichnet werden, durchgeführt werden (Santa-Eulalia u. a., 2011). Diese sind in der Lage, automatisch Pläne zu erzeugen, die die Bedingungen von Aufträgen und Ressourcen berücksichtigen. Gleichzeitig führen sie häufig eine *Optimierung* durch, d.h. sie ermitteln einen Plan, der zu einer Maximierung von Gewinn bzw. zu einer Minimierung von Kosten führt. Sie sind häufig an die betriebswirtschaftliche Verwaltungssoftware des Unternehmens oder der Organisation angekoppelt (ERP-Systeme, ERP = **E**nterprise **R**esource **P**lanning), um von dort die zur Planung benötigten Daten zu beziehen (Staedler u. a., 2012).

Komponenten von Planungssystemen (APS)

Planungssysteme bestehen im Wesentlichen aus den in Abbildung 1.3 gezeigten Komponenten (Turban, Sharda und Delen, 2011; Framinan und Ruiz, 2010; Staedler u. a., 2012; Wierzbicki, Makowski und Wessels, 2000):

- Das *Planungsmodell* enthält die unternehmensspezifischen Regeln und die Bedingungen, die ein gültiger Plan einhalten muss. Ein Produktionsunternehmen legt im Planungsmodell z.B. fest, in welcher Abfolge Ressourcen eingesetzt werden, um bestimmte Produkte herzustellen (Braun und Gruenwald, 2000). Damit spiegelt ein Planungsmodell die strategischen und taktischen Entscheidungen des Unternehmens oder der Organisation wieder (vgl. Abschnitt 1.1.2).

- Die *Datenbankschnittstelle* ermöglicht den Zugriff auf Stammdaten und Auftragsdaten, die in der Datenbank des Planungssystems oder eines angekoppelten ERP-Systems abgespeichert sind.

- Ein Satz von *Lösungsverfahren*, berechnet unter Einhaltung des Planungsmodells einen optimalen Plan zur Durchführung der Aufträge (Braun und Gruenwald, 2000).

- Eine *Benutzerschnittstelle* ermöglicht dem menschlichen Disponenten eine Interaktion mit den restlichen Komponenten, d.h. eine Anpassung des Modells und der Daten sowie die Konfiguration von Lösungsverfahren. Häufig unterstützt die Benutzerschnittstelle auch die Auswahl eines Plans aus einer Liste von Alternativplänen oder die manuelle Anpassung eines Plans.

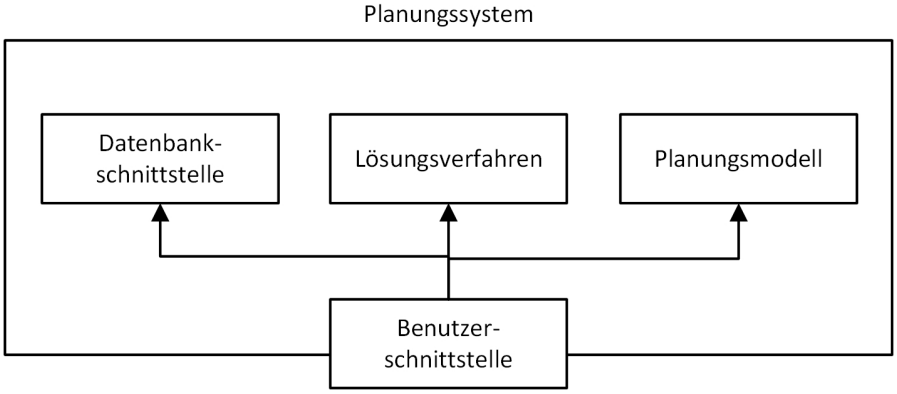


Abbildung 1.3: Komponenten eines Planungssystems

Moderne APS-Systeme sind häufig für einen breiten Anwenderkreis konzipiert, da sie Planungsmodelle enthalten, die allgemein formuliert sind und über zahlreiche Parameter für das unternehmensspezifische Anwendungsgebiet konfiguriert werden können.

Beispiel 1.4: Angabe auf der Webseite von ASPROVA[1], einem Feinplanungssystem für die Produktion:

> „Über 2400 verschiedene Parameter und 4000 Eigenschaften berücksichtigen in Asprova bereits alle erdenklichen Standardsituationen in der Produktion. Dadurch ist es möglich, ohne individuelle Programmierung den aktuellen und zukünftigen Fertigungsprozess in der Software 100 % ig abzubilden bzw. zu simulieren."

Folgende Produkte sind Beispiele für APS-Systeme (vgl. auch Brandenburg u. a. (2013)):

- SAP Advanced Planning and Optimization (SAP APO) als Bestandteil von SAP SCM[2] (Braun und Gruenwald, 2000),
- Infor SCM[3],

[1] Quelle: `http://www.asprova.eu/asprova-kennenlernen/besondere-leistungsmerkmale/abbildung-produktion/`, abgerufen am 26.06.2014

[2] `http://www.sap.com/germany/solution/lob/scm.html`, abgerufen am 26.04.2015

[3] `http://www.infor.com/solutions/scm/`, abgerufen am 19.07.2017

- Preactor 500 APS[4],
- APplus[5],
- ASPROVA APS[6]

Im Folgenden werden ERP- und APS-Systeme unter dem Begriff „ERP-System“ zusammengefasst, da die aktuellen Entwicklungs- und Vermarktungstrends auf alle Arten von Systemen zur Unternehmenssteuerung zutreffen und die Grenzen zwischen ERP und APS zunehmend fließend verlaufen.

Die aktuelle Marktsituation für Planungssysteme

Experten sind der Meinung, dass der Markt für ERP-Software derzeit der größte „Software-Markt nach dem für Betriebssysteme“[7] ist. Bereits jetzt setzen sehr viele Unternehmen ERP-Software ein, auch wenn noch nicht von einer flächendeckenden Verbreitung gesprochen werden kann.
Laut einer 2011 von der Firma Konradin durchgeführten Studie nutzen rund 92% großer industrieller Betriebe mindestens eine ERP-Lösung. Jedoch besitzen nur rund 30 % dieser Anwendungen eine APS-Funktionalität[8] (Konradin, 2011). In der Klasse der kleinen und mittelständischen Unternehmen (KMUs) besitzen lediglich zwei Drittel ein ERP-System[9]. Bei Kleinstunternehmen mit weniger als 10 Mitarbeitern sind es sogar nur 12% (Leyh und Gottwald, 2011). Der Funktionsumfang der eingesetzten Software ist dabei sehr unterschiedlich. In den genannten Studien wurden keine Non-Profit-Organisationen, wie z.B. Krankenhäuser oder Universitäten berücksichtigt.
Anbieter von ERP-Software sind bestrebt,

- Großunternehmen weiterhin als Kunden zu binden und
- kleine und mittelständische Unternehmen als zusätzliche Kunden zu gewinnen.

[4] `http://www.preactor.com/`, abgerufen am 19.07.2017
[5] `https://www.applus-erp.de/`, abgerufen am 19.07.2017
[6] `http://www.asprova.eu/`, abgerufen am 19.07.2017
[7] Newsletter von Messe Stuttgart vom 13.12.2012, `www.messe-stuttgart.de`
[8] in Deutschland, Betriebe ab 50 Mitarbeiter
[9] „Mittelstandsbericht ERP“ der GUS Group (Kiewitt, 2010)

Beide Ziele können laut Analysten nur erreicht werden, wenn die Softwarehersteller auf die steigenden Anforderungen aus der Wirtschaft reagieren (Bayer, 2012). Sie fordern, dass sich die Software schneller an neue Geschäftsstrategien anpassen lässt (Rettig, 2007). Aktuelle Systeme werden als „zu groß, zu langsam, zu komplex und zu teuer“ kritisiert (Nittler, 2007).
Führende Anbieter wie z.B. die SAP AG reagieren darauf mit neuen Technologien und Geschäftsmodellen. Sie setzen zunächst bei dem bisher größten Problem herkömmlicher ERP-Systeme an, nämlich der zeitaufwändigen und komplizierten Installation und Konfiguration. Um diese überflüssig zu machen, wurde das Konzept „Software as a Service (SaaS)“ entworfen, bei dem die Unternehmen über das Internet auf die Software zugreifen, anstatt sie im eigenen Haus zu installieren[10] (Bayer, 2011). Übersetzt heißt es „Software als Dienstleistung“. Dahinter verbirgt sich ein besonderes Bezahlungsmodell: Gezahlt wird in der Regel nur für tatsächlich in Anspruch genommene Dienste, wie z.B. Rechenleistung oder Speicherplatz. Bereits ein Drittel aller betriebswirtschaftlichen Software-Lösungen wird (z.T. neben der herkömmlichen Variante) im SaaS-Modell angeboten (Bayer, 2011). Besonders für kleine und mittelständische Unternehmen ergeben sich daraus große Vorteile: Für sie war die Anschaffung eines ERP-Paketes mitsamt der benötigten IT-Infrastruktur bisher oft unerschwinglich. Das neue Modell ermöglicht vielen Unternehmen erstmals den Zugang zu diesen Angeboten.
Ein weiterer Trend ist die Bereitstellung von ERP-Anwendungen auf Smartphones und Tablets (Hengl, 2011). In Verbindung mit dem SaaS-Konzept ermöglichen mobile Anwendungen Geschäftsentscheidungen zu jeder Zeit und an jedem Ort.

1.1.5 Die Rolle des Disponenten bei der Planung

Als *Disponent* wird in dieser Arbeit der Mitarbeiter bezeichnet, der in einem Unternehmen oder in einer Organisation für die Festlegung operativer Planungsentscheidungen verantwortlich ist. In Produktions- und Logistikunternehmen entspricht dies dem Berufsbild eines Arbeitsplanungsingenieurs[11]:

[10] Man spricht auch davon, dass die Software in der „Cloud“ liegt.

[11] In der englischen Literatur existieren, besonders im Bereich der Produktionsplanung, für den Disponenten die beiden Bezeichnungen „planner“ und „scheduler“. Während der „planner“ hauptsächlich für die Grobplanung zuständig ist, liegt der Arbeitsschwerpunkt des „schedulers“ bei der Feinplanung.

> „Ein Arbeitsplanungsingenieur oder eine Arbeitsplanungsingenieurin entwickeln und gestalten Arbeitsabläufe für die Produktion, den Transport oder die Bearbeitung von Rohstoffen, Agrarprodukten, Versandprodukten und von Industriegütern. Dabei befassen sie sich vor allem mit der Organisation und Koordination sowie mit der Überwachung und Optimierung der Produktions- und Arbeitsabläufe innerhalb ihres Verantwortungsbereiches[12].“

Trotz der Unterstützung durch Softwaresysteme spielt der Disponent bei der Planung eine wichtige Rolle. Dies trifft besonders auf die Feinplanung zu, da hier der Disponent[13] die Person ist, welche die betrieblichen Gegebenheiten vor Ort genau kennt und die praktische Umsetzbarkeit von Plänen einschätzen kann. Häufig trifft er Planungsentscheidungen, die aus seinem Expertenwissen resultieren und eine reibungslose Auftragsdurchführung sowie die Zufriedenheit der Mitarbeiter fördern. Diese wären in der Regel bei einer rein automatischen Planerstellung nicht berücksichtigt worden.

Aktivitäten des Disponenten

Um auf dem neuesten Stand zu bleiben, verbringt der Disponent in der Regel viel Zeit damit, die Planausführung zu beobachten (Jackson, Wilson und MacCarthy, 2004; McKay und Buzacott, 2000). Dabei sucht er nach bestimmten Hinweisen, die erfahrungsgemäß auf eine Instabilität hindeuten (McKay und Buzacott, 2000). Sein Ziel ist es, potenzielle Engpässe, Verzögerungen oder andere Störungen frühzeitig zu erkennen. Er versucht im Fall einer Störung durch eine kurzfristige Anpassung des Feinplans sicherzustellen, dass die Unternehmensziele, wie z.B. pünktliche Lieferung und effizienter Einsatz der Ressourcen, trotzdem weitestgehend eingehalten werden können (Wiers, 1997a; Crawford u. a., 1999; Portougal und Robb, 2000; Jackson, Wilson und MacCarthy, 2004). Eine Übersicht über Störungen, die speziell in der Produktion auftreten können, ist bei Stoop und Wiers (1996) zu finden. Störungen, die zur Unterbrechung der Planausführung oder zu Abweichungen vom

[12] Quelle (abgerufen am 28.06.2014): `http://www.gehaltsvergleich.com/stellenanzeigen/arbeitsplanungsingenieur-arbeitsplanungsingenieurin.html`,

[13] In dieser Arbeit wird nur ein einzelner Disponent betrachtet. Der Informationsaustausch zwischen mehreren Disponenten (Snoo u. a., 2011), die z.T. auch auf unterschiedlichen Unternehmensebenen arbeiten, gehört nicht zum Forschungsgegenstand.

Plan führen, können wirtschaftliche Verluste zur Folge haben. Der Disponent ist daher auch gefordert, negative Auswirkungen von Störungen durch eine vorausschauende Planung zu vermeiden (McKay und Wiers, 1999; Morikawa und Takahashi, 2006). Er kann z.B. die Durchführung eines Auftrags, bei dem die Überschreitung des Fälligkeitstermins vom Kunden nicht akzeptiert wird, so zeitig wie möglich einplanen.
Des Weiteren ist der Disponent während der Planung gefordert, die Unsicherheit, der die vorliegenden Auftragsdaten unterliegen, durch seine Erfahrung auszugleichen. Die Unsicherheit ist z.B. hoch, wenn die verplanten Aufträge häufig vom Kunden wieder zurückgezogen werden oder wenn sich Fälligkeitstermine und Bestellmengen häufig ändern. Der Disponent kann in diesem Fall z.B. Leerlaufzeiten von Maschinen nutzen, um zusätzliche Mengen eines Produktes herzustellen. Damit ist das Unternehmen in der Lage, auf eine kurzfristige Erhöhung der Nachfrage zu reagieren.
Durch sein Wissen nimmt der Disponent eine wichtige Vermittlerfunktion im Unternehmen ein (Cegarra und Wezel, 2011c). Häufig ist es seine Aufgabe, Kompromisse zwischen den teilweise widersprüchlichen Interessen verschiedener Abteilungen herbeizuführen. So verteidigt er z.B. die Interessen der Werkstattarbeiter gegenüber dem Management, wenn Verzögerungen bei der Auftragsausführung begründet werden müssen.

Beispiel 1.5: In einem Produktionsunternehmen wird in einem Meeting gemeinsam mit dem Disponenten die Planung für die nächste Woche diskutiert. Die Abteilung für Kundenbetreuung möchte kurze Lieferzeiten aushandeln, während die Werkstattarbeiter für einen geglätteten Produktionsablauf mit wenigen Rüstvorgängen plädieren.

Planung im sozio-technischen System

In vielen Bereichen ist es unerlässlich, dass sich ein menschlicher Disponent an der Planerstellung beteiligt, auch wenn diese softwaregestützt abläuft. Ein Plan wird also nicht von einem rein technischen, sondern von einem *sozio-technischen System* erstellt, in dem soziale und technische Komponenten miteinander interagieren (Gasser, Fischer und Wäfler, 2011). In einem sozio-technischen System bilden beide Komponenten eine Einheit, um ein Problem zu lösen oder ein Ergebnis zu produ-

zieren und kommunizieren miteinander über eine Mensch-Maschine-Schnittstelle (Davis u. a., 2014).
Bei einer Änderung der technischen Komponente ändert sich auch das Zusammenspiel mit der sozialen Komponente und damit die Funktionsweise des Gesamtsystems. Dadurch wird die Qualität der Ergebnisse beeinflusst, die vom Gesamtsystem produziert werden. Eine Vernachlässigung der sozialen Komponente, wie z.B. des Disponenten, kann dazu führen, dass der (wirtschaftliche) Erfolg bei der Einführung oder Erneuerung des technischen Systems, wie z.B. einer Planungssoftware, ausbleibt oder sich verschlechtert (Wäfler u. a., 2011).

1.2 Nutzerakzeptanz von Planungssystemen

Der erfolgreiche Einsatz eines technischen Systems, wie z.B. einer Planungssoftware, hängt entscheidend von seiner Akzeptanz durch die Anwender ab. Eine hohe Akzeptanz bedeutet, dass das System effektiv und in der vorgesehenen Art und Weise eingesetzt wird (Garača, 2011). Eine niedrige Akzeptanz führt dazu, dass der Einsatz des Systems ganz oder teilweise vermieden und stattdessen auf andere Hilfsmittel, wie z.B. papierbasierte Planung oder Excel-Tabellen, zurückgegriffen wird (Wäfler u. a., 2011). Die Kosten für die Anschaffung des technischen Systems in einem Unternehmen werden dabei nicht durch den finanziellen Nutzen seines Einsatzes gedeckt (Amoako-Gyampah, 2007). Gleichzeitig bleiben viele Vorteile der Datenverwaltung, wie z.B. eine zentrale und gesicherte Datenspeicherung, ungenutzt, da die individuellen Hilfsmittel in der Regel von der betrieblichen Dateninfrastruktur entkoppelt sind. Bei der Entwicklung und Einführung von Planungssystemen muss daher die Akzeptanz der Software durch die Disponenten sichergestellt werden.
In diesem Abschnitt wird zunächst untersucht, welche Faktoren die Nutzerakzeptanz von Planungssystemen beeinflussen. In Abschnitt 1.2.2 werden Defizite identifiziert, die Planungssysteme bezüglich dieser Faktoren aufweisen. Anschließend werden Forschungsdisziplinen vorgestellt, die involviert werden müssen, um langfristig Konzepte für Planungssysteme mit einer höheren Nutzerakzeptanz entwickeln zu können.

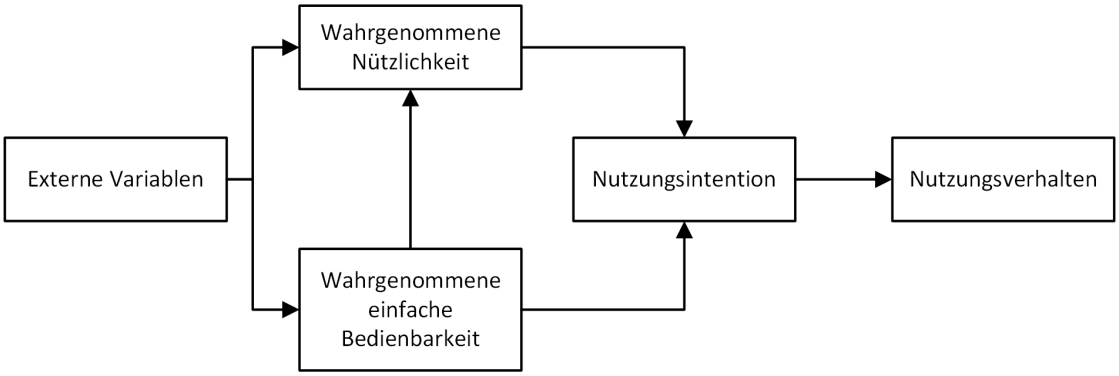


Abbildung 1.4: Einfaches Technologieakzeptanzmodell nach (Davis, 1993)

1.2.1 Einflussfaktoren auf die Nutzerakzeptanz von Planungssystemen

Zur Untersuchung der Einflussfaktoren auf die Akzeptanz von Technologien, insbesondere von Informationssystemen, hat sich das *Technologieakzeptanzmodell (TAM)* bewährt (Davis, 1993). Nach diesem Modell wird ein Nutzungsverhalten, welches einer hohen Akzeptanz entspricht, durch eine positive Nutzungsintention hervorgerufen. Diese wird wiederum durch die Faktoren „Wahrgenommener Nutzen“ und „Wahrgenommene einfache Bedienbarkeit“ beeinflusst (Abbildung 1.4). Empirische Studien bestätigen die Gültigkeit der Abhängigkeiten zwischen diesen grundlegenden Faktoren für ERP-Systeme (Calisir und Calisir, 2004). Die Beobachtungen von Calisir und Calisir (2004) und Morris und Dillon (1997) deuten zudem darauf hin, dass der wahrgenommene Nutzen einen größeren Einfluss auf die Akzeptanz besitzt als die wahrgenommene einfache Bedienbarkeit. Um das Zustandekommen von Akzeptanz zu erklären, wird in empirischen Studien häufig der Einfluss von weiteren externen Faktoren, wie z.B. der Grad der Einbeziehung von Anwendern in den Entwicklungsprozess, gemessen (Amoako-Gyampah, 2007; Shih und Huang, 2009). Im Folgenden werden die wesentlichen Faktoren des Akzeptanzmodells näher erläutert.

Wahrgenommene einfache Bedienbarkeit

Die wahrgenommene einfache Bedienbarkeit wird definiert als „der Grad, in dem eine Person glaubt, dass die Verwendung eines bestimmten Systems frei von Aufwand ist“ (Davis, 1989). Eine Software ist einfach bedienbar, wenn sie selbster-

klärend, leicht zu erlernen, kontrollierbar, flexibel und fehlertolerant ist und ein schnelles Arbeiten ermöglicht. Das Konzept ist damit weitestgehend identisch mit dem Konzept der *Gebrauchstauglichkeit* (engl. usability) einer Software (Morris und Dillon, 1997). Merkmale gebrauchstauglicher Benutzeroberflächen werden in der DIN-Norm DIN EN ISO 9241-110 (Schneider, 2008) definiert.
Nach Nielsen und Giluz (2007) und Marcus (2005) lohnt sich für ein Unternehmen die Investition in die einfache Bedienbarkeit von Software, da sie den Schulungsaufwand senkt und die Produktivität der Mitarbeiter erhöht. Auch kleine Unternehmen, die es sich nicht leisten können, ihre Mitarbeiter auf speziellen Seminaren zu schulen, profitieren von einer selbsterklärenden Software. Das gleiche gilt für Unternehmen, die Software im SaaS-Modell über das Internet ohne persönlichen Kontakt mit Beratern nutzen (Ryan, 2009).
Darüber hinaus sind die Ansprüche der Anwender an die Bedienbarkeit von ERP-Software gestiegen, da für sie der Umgang mit Software zu privaten Zwecken selbstverständlich geworden ist (Galer, 2012). Eine einfache Bedienbarkeit hat daher einen positiven Einfluss auf die Zufriedenheit mit ERP-Software, wodurch die Nutzungsintention erhöht wird (Garača, 2011).
Insgesamt spielt eine einfache Bedienbarkeit eine immer größere Rolle bei der Entscheidung für oder gegen die Anschaffung eines (neuen) ERP-Systems (Woywode u. a., 2012; Sontow, Treutlein und Sontow, 2014).

Wahrgenommene Nützlichkeit

Der wahrgenommene Nutzen oder die wahrgenommene Nützlichkeit ist „der Grad, in dem ein Anwender glaubt, dass die Verwendung des Systems seine oder ihre Leistung steigern wird“ (Davis, 1989). Im Technologieakzeptanzmodell steht also nicht die Nützlichkeit in Bezug auf das Erreichen von Unternehmenszielen, sondern die Nützlichkeit für den Anwender im Vordergrund. Sie wird von ihm als hoch empfunden, wenn ihm die Software dabei hilft, innerhalb des Unternehmens Anerkennung für eine gute Arbeitsleistung zu erlangen. Auch die wahrgenommene einfache Bedienbarkeit kann die Nützlichkeit positiv oder negativ beeinflussen (siehe Abbildung 1.4).
Welche konkreten Eigenschaften eine Software besitzen muss, um eine hohe wahrgenommene Nützlichkeit zu erzielen, muss jeweils für den konkreten Anwendungsfall ermittelt werden. Bei einer Planungssoftware kann davon ausgegangen werden,

dass der Disponent die Software als nützlich empfindet, wenn sie ihn dabei unterstützt, die in Abschnitt 1.1.4 beschriebene Rolle auszufüllen, d.h. durchführbare Pläne zu erstellen und Probleme zu erkennen und zu beseitigen (Cegarra und Wezel, 2012). Dies entspricht dem Ziel, in einem optimal funktionierenden soziotechnischen System zu arbeiten, welches auch im Interesse der Unternehmensleitung ist.
Um mit dem Disponenten zu kooperieren, muss eine Planungssoftware Interaktionsmöglichkeiten zur Eingabe und Ausgabe von Informationen an der Mensch-Computer-Schnittstelle bereitstellen. Die Defizite, die Planungssysteme bei der Bereitstellung geeigneter Interaktionsmöglichkeiten aufweisen, werden im nächsten Abschnitt diskutiert.

1.2.2 Bewertung von Planungssystemen nach Kriterien der Nutzerakzeptanz

Im Folgenden werden Planungssysteme nach den TAM-Kriterien der wahrgenommenen einfachen Bedienbarkeit und des wahrgenommenen Nutzens bewertet. Sie sollen Aufschluss darüber geben, ob moderne Planungssysteme in der Lage sind, den Disponenten bei seinen Aufgaben zu unterstützen.

Wahrgenommene einfache Bedienbarkeit

Experten, wie z.B. Heikki Topi von der Bentley Universität (Bjorlin, 2010) bescheinigen ERP-Systemen einen Mangel an Selbstbeschreibungsfähigkeit. Verschiedene Studien haben eine Reihe von Bedienbarkeits-Problemen festgestellt, die den effizienten Einsatz der Systeme erschweren (Oja und Lucas, 2010; Osintsev, 2012; Topi, Lucas und Babaian, 2005). Auch die Anwender selbst stellen diese Mängel fest: In einer weltweiten Studie fand das IFS Nordamerika heraus, dass nur ein Drittel aller ERP-Kunden die Software für intuitiv und einfach benutzbar hält (Matthews, 2008). Aus diesem Grund werden ERP-Systeme häufig nur widerwillig eingesetzt oder von den Mitarbeitern umgangen. Laut einer weiteren, im Jahr 2011 durchgeführten Studie des IFS steigen ca. 50% der Anwender auf alternative Werkzeuge wie z.B. Microsoft Excel um, wenn die Planungssoftware zu umständ-

lich zu bedienen ist[14] (IFS, 2011). Dieses Ergebnis veranlasste IFS sogar zu der (nicht ganz ernst gemeinten) Annahme, der Begriff ERP stünde für „**E**xcel **R**uns **P**roduction“ (Excel steuert die Produktion). Die Verbesserung der Bedienbarkeit ist daher ein aktueller Schwerpunkt bei der Entwicklung von Planungssystemen (vgl. Bjorlin (2012), King (2012)).

Wahrgenommene Nützlichkeit

Es können keine allgemeingültigen Studien angeführt werden, die sich mit der Frage beschäftigen, ob die wahrgenommene Nützlichkeit von modernen Planungssystemen ausreichend ist und ob sie dem Disponenten geeignete Interaktionsmöglichkeiten bieten. Wie im Folgenden gezeigt wird, sind sich jedoch eine Reihe von Wissenschaftlern darüber einig, dass die bisherigen theoretischen Konzepte zur Gestaltung von Planungssystemen den Nutzen für den Anwender stark vernachlässigen.
Laut (Cegarra und Wezel, 2011b) wird bei der Konzeption von Planungssystemen alles, was moderne Technologien zu bieten haben, verwendet, um so viele Planungsentscheidungen wie möglich zu automatisieren. Übrig gebliebene Aufgaben eines Planungsprozesses werden dem Disponenten zugewiesen. Dazu gehört z.B. das Einstellen von Parametern des Lösungsverfahrens oder die nachträgliche Modifikation eines automatisch erstellten Plans. In der Theorie beschränkt sich die Einbeziehung des Disponenten also darauf, ihm eine Überwachung des Planungsprozesses und, falls notwendig, eine Korrektur der Ergebnisse zu ermöglichen.
Ein Eingriff in den Planerstellungsprozess ist in der Praxis jedoch viel häufiger erforderlich als in der Theorie angenommen. Laut Jackson, Wilson und MacCarthy (2004) liegt den meisten Planungssystemen die Annahme zugrunde, dass „die Situation in der Werkstatt stabil ist, alle Materialien verfügbar sind, die Arbeiter in der Lage sind, ihre Aufgaben auszuführen und keine Maschinen ausfallen“. Diese Annahme führt zu stark vereinfachten Planungsmodellen (Abschnitt 1.1.4), die die Realität im Unternehmen nicht immer ausreichend abbilden (MacCarthy, Wilson und Crawford, 2001; Cegarra und Wezel, 2012). Sobald die Realität vom Planungsmodell abweicht, sind die vom Computer generierten Pläne hinfällig. Der Disponent, der die Verantwortung für die Erstellung durchführbarer Pläne trägt,

[14] In die Studie wurden folgende Arten von Systemen einbezogen: „Enterprise Resource Planning“ (ERP, hier als Oberbegriff für ERP und APS), „Customer Relationship Management“ (CRM, engl. Kundenbeziehungsmanagement), „Enterprise Asset Management“ (EAM, engl. Inventarmanagementsystem), „Business Intelligence“ (BI).

muss a) diese Situation erkennen und b) die fehlenden Informationen in die Planung einbringen.
Planungssysteme, die nach dem hier beschriebenen Prinzip konzipiert wurden, bieten ihm keinerlei Unterstützung bei der Suche nach Entscheidungen, mit denen Pläne angepasst und Probleme beseitigt oder vermieden werden können (Gasser, Fischer und Wäfler, 2011). Um die Konsequenzen und Risiken möglicher Planungsentscheidungen gegeneinander abzuwägen, benötigt der Disponent Informationen, die ihm das Planungssystem aufgrund konzeptioneller Mängel häufig nicht liefern kann. Sie müssen daher aus anderen Informationsquellen bezogen oder mit Hilfe von externen Werkzeugen ermittelt werden (Berglund und Karltun, 2007).
Wenn sich die Beteiligung des Disponenten an der Planung auf die Korrektur von Plänen beschränkt, verschlechtert sich u.U. sein Expertenwissen über wichtige betriebliche Zusammenhänge. Es fällt ihm u.U. zunehmend schwerer, die automatisch generierten Pläne einzuschätzen und er akzeptiert diese häufiger, ohne sie zu hinterfragen (Wiers, 1997a) (dies wird auch als *Complacency-Effekt*[15] bezeichnet Cegarra und Hoc (2008) und Parasuraman und Manzey (2010)).
Um die Nützlichkeit von Planungssystemen zu erhöhen, müssen laut Gasser, Fischer und Wäfler (2011) die kognitiven Aktivitäten des Disponenten, wie z.B. Vorgänge zur Entscheidungsfindung und Problemlösung, in Zukunft vom Planungssystem direkt unterstützt werden:

> „Wenn durch eine bessere Anpassung des Systementwurfs an die Arbeitsweise des Experten mehr unterstützende Merkmale geschaffen werden, wird daraus höchstwahrscheinlich eine leistungsfähigere Planung und Disposition resultieren. Um dieses Ziel zu erreichen, ist immer noch ein großer Forschungs- und Entwicklungsaufwand erforderlich."

Diese Forderung bezieht sich auf die Gestaltung der Mensch-Computer-Schnittstelle von Planungssystemen (Gasser, Fischer und Wäfler, 2011; Riedel u. a., 2011) und auf die Entwicklung automatischer Lösungsverfahren zur Unterstützung des Disponenten (Wiers, 1997a; Cegarra und Wezel, 2011b). Der Disponent sollte nicht nur

[15] Als Complacency-Effekt (zu deutsch: Selbstgefälligkeit) wird im Allgemeinen ein übersteigertes Vertrauen in die Automatisierung bezeichnet, das dazu führt, dass der Mensch das System nur noch unzureichend kontrolliert (Parasuraman und Manzey, 2010). Laut Cegarra und Hoc (2008) kann der Complacency-Effekt speziell bei Planungsproblemen auch daraus resultieren, dass die Überprüfung und/oder Korrektur von Plänen mit einem zu hohen Arbeitsaufwand verbunden ist.

die Unzulänglichkeiten eines Planungssystems kompensieren müssen, welches ansonsten als Entscheidungsautorität fungiert (Cegarra und Wezel, 2011c). Stattdessen sollte das Planungssystem vom Disponenten als nützliches Werkzeug zur Planerstellung betrachtet werden. Wenn die Entscheidungsautorität wieder stärker auf die Seite des Disponenten verlagert wird, kann vermutlich auch der Complacency-Effekt reduziert werden.

1.2.3 Forschungsgebiete zur Gestaltung von Planungssystemen mit hoher Akzeptanz

Die Anforderung, Planungssysteme mit einer optimalen Unterstützung für den Disponenten zu gestalten, eröffnet ein interdisziplinäres Forschungsgebiet. Die unterschiedlichen Disziplinen, die in die Forschung einbezogen werden müssen, wurden von MacCarthy, Wilson und Crawford (2001) in einem Rahmenkonzept festgehalten (Abbildung 1.5).
Den Ausgangspunkt bildet stets die Betrachtung des *organisatorischen Kontexts*, bei der auch ein Verständnis von den *betrieblichen Arbeitsabläufen* gewonnen werden muss. Der Kontext wird bestimmt durch die unternehmensspezifische Ausprägung des Planungsproblems (z.B. Produktions-, Flotten-, oder Schichtplanung) und die damit verbundenen Anforderungen an die Planerstellung. Letztere erfolgt in einem *Planungsprozess*, der von entsprechend ausgebildeten *Disponenten* und mit Hilfe von *Planungssystemen* ausgeführt wird (sozio-technisches System). Es ist erforderlich, den Planungsprozess im Kontext des sozio-technischen Systems und des Planungsproblems zu erforschen, um Rückschlüsse für seine Verbesserung und Erleichterung ziehen zu können. Entsprechende Feldstudien wurden z.B. von Gasser, Fischer und Wäfler (2011), McKay und Buzacott (2000), Usher und Kaber (2000) (Produktionsplanung), Mietus (1994) (Schichtplanung im Krankenhaus), Wong und Bl (2002) (Flottenplanung für Notfallkrankenwagen) und Rahimi und Dessouky (2001) (Flottenplanung für den Behindertenfahrdienst) durchgeführt. Weitere Feldstudien werden von MacCarthy (2006) genannt.
Kenntnisse über den praktischen Ablauf der Planung führen zu einem besseren Verständnis der Planungsziele und der Anforderungen an die Planung. Es können Kriterien für die *Leistungsmessung*, d.h. für die Einschätzung, ob ein Planungsprozess erfolgreich ist, entwickelt werden. Eine umfangreiche Sammlung von Leistungskriterien für eine große Bandbreite an Planungsproblemen wurde von Snoo,

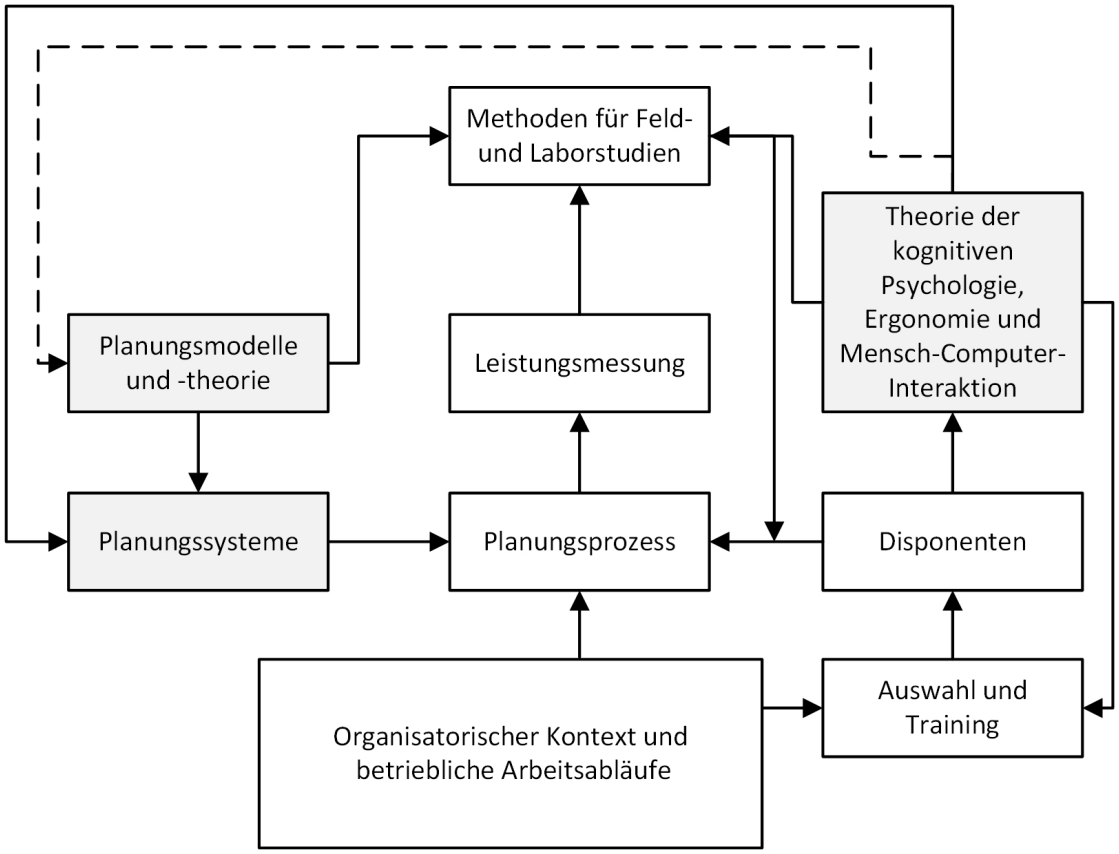


Abbildung 1.5: Rahmenkonzept für die Erforschung der Rolle des Disponenten bei der Planung nach MacCarthy, Wilson und Crawford (2001). Grau hinterlegte Disziplinen werden in dieser Arbeit betrachtet.

Wezel und Jorna (2011) zusammengestellt.

Die *Methoden*, nach der Feld- und Laborstudien durchgeführt werden, müssen ebenfalls weiterentwickelt werden, damit mit ihrer Hilfe gut verwendbare Informationen gewonnen werden können. Forschungsbeiträge auf diesem Gebiet wurden u.a. von Cegarra (2008) (beliebige Anwendungsbereiche), Crawford u. a. (1999), Cegarra und Wezel (2011a) und Morikawa und Takahashi (2006) (Produktionsplanung) geliefert.

Ein weiteres Forschungsgebiet sind mathematische und algorithmische Theorien zur Lösung und Modellierung von Planungsproblemen. Sie sind die Grundlage zur Entwicklung von Planungssoftware, mit der Pläne automatisch oder teilweise automatisch erstellt werden können. Ein Standardwerk wurde dazu z.B. von Pinedo (2012) geschaffen.

Die Ergebnisse der Feldstudien können für einzelne Anwendungsfälle direkt in An-

forderungen zur Gestaltung der Mensch-Computer-Schnittstelle in Planungssystemen übersetzt werden (wie z.B. bei McKay und Buzacott (2000), McKay und Wiers (2002) für die Produktionsplanung). Um allgemeingültige Konzepte zur Erhöhung der Bedienbarkeit und Nützlichkeit von Planungssystemen zu entwickeln, müssen in die Anforderungen jedoch auch die Grundlagen der *kognitiven Psychologie, Ergonomie* und *Mensch-Computer-Interaktion* einfließen. Diese Disziplinen haben bisher einen zu geringen Einfluss ausgeübt und müssen in Zukunft stärker involviert werden. Anforderungen an Planungssysteme unter Einbeziehung der kognitiven Psychologie bzw. Mensch-Computer-Interaktion wurden u.a. von Cegarra und Wezel (2012) (beliebige Anwendungsbereiche), Igbaria u. a. (1996), Gayialis und Tatsiopoulos (2004) (Flottenplanung), Wiers und Van Der Schaaf (1997), Higgins (1999), Higgins (2001) und Wiers (1997b) (Produktionsplanung) formuliert.

Der Forschungsstand muss innerhalb einer Disziplin überprüft und ggf. aktualisiert werden, wenn neue Erkenntnisse in den beeinflussenden Disziplinen vorliegen. Werden z.B. neue Planungssysteme entwickelt, so beeinflussen diese den Planungsprozess. Dessen Erfolg, d.h. dessen Effektivität und Effizienz, muss im Rahmen von Feldstudien evaluiert werden. Auch die Methodik der Feld- und Laborstudien muss ggf. angepasst werden, sobald neue Erkenntnisse bezüglich der Leistungsmessung, der Mensch-Computer-Interaktion oder der Planungstheorie vorliegen.

Obwohl das Rahmenkonzept unter der Betrachtung von Produktionsplanungsproblemen entstanden ist, kann es, wie die angeführten Forschungsbeiträge zeigen, auch für viele weitere Planungsprobleme angewendet werden. Ein Bedarf nach Planungssystemen mit einer guten interaktiven Unterstützung des Disponenten wurde auch im Bereich der Universitätsstundenplanung festgestellt. So zählen die Gestaltung der Benutzerschnittstelle und die Entwicklung von Methoden zur Interaktion mit dem Nutzer für McCollum (2006) zu den zukünftigen Herausforderungen bei der Lösung komplexer, realistischer Stunden- und Prüfungsplanungsprobleme (vgl. auch McCollum (2007)). Ebenso wurde im Bereich der Schichtplanung im Krankenhaus laut Burke u. a. (2004) bisher zu wenig Forschungsaufwand in die Einbeziehung von Nutzereingaben in den Planungsprozess investiert.

1.3 Modelle der Funktionsaufteilung zwischen Mensch und Computer

Wissenschaftler gehen davon aus, dass moderne Planungssysteme Defizite bezüglich der Nützlichkeit aufweisen (siehe Abschnitt 1.2.2). Es kann vermutet werden, dass die zur Gestaltung von Planungssystemen bereitgestellten Konzepte der Mensch-Computer-Interaktion, insbesondere der Funktionsaufteilung, unzureichend und damit die Ursache für diese Defizite sind. Nach einer Begriffsdefinition werden in diesem Abschnitt bestehende Konzepte der Funktionsaufteilung untersucht und anschließend bewertet.

1.3.1 Definition der Funktionsaufteilung und ihr Einfluss auf die wahrgenommene Nützlichkeit

Die *Funktionsaufteilung* einer Software legt fest, welche Arbeitsaufgaben jeweils von Mensch und Computer übernommen werden, um ein bestimmtes Arbeitsziel zu erreichen. Außerdem wird durch die Funktionsaufteilung geregelt, in welcher Abfolge Mensch und Computer miteinander interagieren (Pritchett, Kim und Feigh, 2014; Wäfler, 2001).
Die Funktionsaufteilung steht in direktem Zusammenhang mit dem Grad der *Automatisierung*. Wird ein Großteil der Arbeitsaufgaben vom Anwender manuell erledigt, so liegt ein niedriger Automatisierungsgrad vor. Je mehr Arbeitsschritte vom Computer übernommen werden, desto höher ist der Automatisierungsgrad der Software. Die Funktionsaufteilung und der damit verbundene Automatisierungsgrad können die Nutzerakzeptanz positiv oder negativ beeinflussen. So kann z.B. die Automatisierung von Arbeitsaufgaben, die für den Anwender mit einem hohen Aufwand oder einer hohen Schwierigkeit verbunden sind, den wahrgenommenen Nutzen der Software steigern. Auf der anderen Seite kann ein hoher Automatisierungsgrad die Akzeptanz beeinträchtigen, wenn aufgrund der Automatisierung die Entscheidungen des Anwenders nicht mehr berücksichtigt werden und somit das Ergebnis von seinen Vorstellungen abweicht (Wäfler, 2001).
Einige Modelle der Funktionsaufteilung betrachten Mensch und Computer als getrennte Ausführungseinheiten (z.B. bei Parasuraman, Sheridan und Wickens (2000)). Es wird empfohlen, ihnen jeweils die Aufgaben zuzuweisen, für die sie

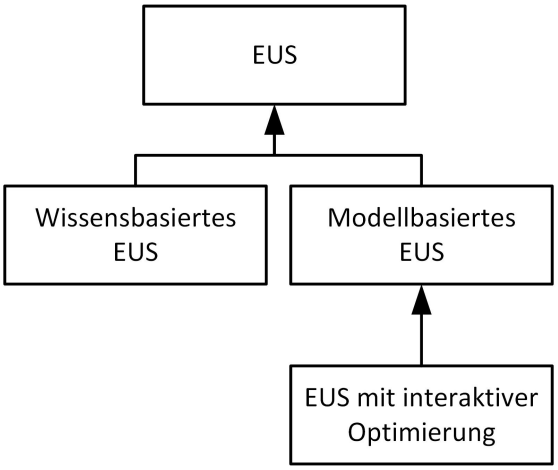


Abbildung 1.6: Spezialisierungen von EUS mit unterschiedlichen Arten der Funktionsaufteilung

aufgrund ihrer Fähigkeiten zur Informationsverarbeitung am besten geeignet sind (Dekker und Woods, 2002). Ein Mensch-Computer-Vergleich nach Kriterien der Rechenleistung fällt jedoch aufgrund der kognitiven Einschränkungen des Menschen in den meisten Fällen zugunsten des Computers aus (Wäfler, 2001). Daher führt die Trennung von Mensch und Computer nach Kantowitz, B. H. und Sorkin, R. D. (1987) (Cegarra und Wezel, 2011b) unweigerlich zu einer Verlagerung des Arbeitsschwerpunktes auf die Seite des Computers. Übrig gebliebene, nicht automatisierbare Aufgaben zur Überwachung und Störungsbehandlung werden dem Menschen überlassen (Endsley, 1999). Die Folgen sind im Bereich der Planung häufig eine mangelnde Entscheidungsunterstützung sowie der Complacency-Effekt (vgl. Abschnitt 1.2.2).
Neue Konzepte der Funktionsaufteilung fordern, dass Mensch und Computer kooperativ zusammenarbeiten und sich bei jeder Aufgabe gegenseitig ergänzen (Hoc, 2000; Cummings, 2004; Parasuraman und Manzey, 2010). Ein technisches System soll dazu dienen, „menschliche Stärken zu fördern und menschliche Schwächen zu kompensieren“ (Wäfler, 2001). Auf diese Weise sollen bessere Ergebnisse erzielt werden, als wenn beide Parteien getrennt voneinander arbeiten.

1.3.2 Modelle der Funktionsaufteilung für die Planung

Im Folgenden werden Modelle der Funktionsaufteilung vorgestellt, die auf dem Gebiet der Planung weit verbreitet sind. Zunächst wird das übergeordnete Modell der

Entscheidungsunterstützungssysteme (EUS) beschrieben. Es gibt zahlreiche Kategorien von EUS (vgl. Burstein und Holsapple (2008), Turban, Sharda und Delen (2011)). Modellbasierte und wissensbasierte EUS sind im Bereich der Planung häufig anzutreffen und werden daher im zweiten Abschnitt näher vorgestellt. Darauf aufbauend wird im letzten Abschnitt das Konzept der interaktiven Optimierung für modellbasierte EUS vorgestellt, welches eine weitergehende Einbeziehung des menschlichen Experten vorsieht (Abbildung 1.6).

Das Konzept von EUS

Probleme können vom Computer automatisch gelöst werden, wenn sie vollständig strukturiert bzw. programmierbar sind. In diesem Fall liegt das zur Lösung erforderliche Wissen kodiert vor (z.B. in Form von mathematischen Modellen) und die Beteiligung eines menschlichen Experten ist zur Problemlösung nicht erforderlich. Bei einem semi-strukturierten Problem ist das zur Lösung erforderliche Wissen dagegen nicht vollständig kodiert oder kodierbar. Aus diesem Grund erfüllen u.U. nicht alle Lösungen, die auf der Basis des Modells erzeugt werden, die Anforderungen des menschlichen Experten. Eine rein manuelle Problemlösung ist in der Regel trotzdem nicht erforderlich, da die unvollständigen Modelle vom Experten genutzt werden können, um eine Reihe von Alternativlösungen zu erzeugen. Sein unstrukturiertes Zusatzwissen lässt er dabei sowohl in die Erzeugung, als auch in die Bewertung der Lösungen einfließen. Schließlich wählt er die aus seiner Sicht am besten geeignete Lösung aus. Der Prozess der Erzeugung, Bewertung und Auswahl von Lösungen wird auch als Entscheidungsprozess bezeichnet. Gorry und Morton (1971) definieren Systeme zur Lösung semi-strukturierter Probleme als Entscheidungsunterstützungssysteme (EUS):

> „Systeme auf diesem Gebiet müssen die Entwicklung der Entscheidungsfähigkeit des Managers unterstützen, indem sie sein Verständnis von der Umgebung erhöhen. Die Rolle von Entscheidungsunterstützungssystemen ist eine erkenntnisvermittelnde. Auch wenn der Entscheidungsprozess nicht strukturiert werden kann, können wir Modelle der Umgebung bereitstellen, aus denen der Manager Einsichten über die Zusammenhänge seiner Entscheidungen mit den Zielen, die er verfolgt, gewinnen kann.“

EUS helfen also dem menschlichen Experten dabei, mögliche Entscheidungen hinsichtlich der Unternehmensziele zu beurteilen (vgl. auch Turban, Sharda und Delen (2011) und Burstein und Holsapple (2008)). In der Regel ist das strukturierte Wissen aufgrund seines Umfangs und seiner Komplexität dem menschlichen Experten nur teilweise bekannt. Daher muss stets überprüft werden, ob die Entscheidungen, die aus unstrukturiertem Wissen resultieren, auch unter Berücksichtigung des strukturierten Wissens umsetzbar bzw. akzeptabel sind. Der Experte kann mit Hilfe des EUS z.B. zur Einsicht kommen, dass seine Entscheidung zu einer Lösung mit schlechten Kosten- oder Leistungskennzahlen führt und daher abgewandelt werden muss. Es ist z.B. folgendes Szenario denkbar:

1. Unstrukturiertes Wissen führt dazu, dass der Experte eine Alternativlösung mit Eigenschaft X vom Computer erzeugen lässt. Dabei wird das strukturierte Wissen berücksichtigt.

2. Der Experte bewertet die erzeugte Alternativlösung. Dabei können sowohl die vom Computer ermittelten strukturierten Bewertungsfunktionen (wie z.B. Kosten- und Leistungskennzahlen) als auch weitere unstrukturierte Bewertungskriterien des Experten einfließen.

3. Die Bewertung kann z.B. zu der Schlussfolgerung führen, dass Eigenschaft X im Zusammenspiel mit den restlichen, strukturierten Bedingungen nicht zu einer akzeptablen Lösung führt.

4. Der Experte kann sich weitere Alternativlösungen erzeugen lassen, bevor er sich für eine Lösung entscheidet.

Wissensbasierte und modellbasierte EUS für die Planung

In *wissensbasierten EUS* wird die Entscheidungslogik eines menschlichen Experten hinterlegt. Das Wissen kann z.B. in Form von Wenn-Dann-Regeln oder semantischen Netzen repräsentiert werden. Bei der Problemlösung simuliert das System in abstrakter Form die menschliche Denkweise und ist damit in der Lage, dem Anwender den Lösungsweg in der von ihm gewohnten Sprache zu erklären. Die Arbeitsweise des Systems ist somit transparent und nachvollziehbar, wodurch das Vertrauen des Anwenders in die Systementscheidungen gefördert wird (Klein und Methlie, 2009).

Ein wissensbasiertes EUS dient gemäß der allgemeinen Philosophie von EUS dazu, die Entscheidungsfähigkeit des Anwenders zu stärken, anstatt ihm die Entscheidungen abzunehmen. Daher sind in der Regel Möglichkeiten zur Auswahl von Lösungsstrategien oder zur Priorisierung von Regeln vorgesehen. Beispiele für wissensbasierte EUS in der Produktionsplanung sind bei Oezbayrak und Bell (2003) und Klein, Lecomte und Dejax (1993) zu finden.
In *modellbasierten EUS* werden die Eigenschaften von Entitäten der Anwendungsdomäne und die Beziehungen zwischen ihnen mathematisch formuliert (z.B. die Kapazitäten von Maschinen oder die Reihenfolge von Aufgaben, (Wezel, 2006)). Lösungsverfahren, die z.B. aus dem Gebiet des Operations Research stammen, können Lösungen erzeugen, die keine im Modell vorgegebenen Bedingungen verletzen. Darüber hinaus wird es dem Anwender ermöglicht, die Eigenschaften der Lösungen mitzubestimmen, indem Möglichkeiten zur Änderung von Parametern bereitgestellt werden. Das EUS aktualisiert nach jeder Änderung die Lösung und berechnet die Auswirkungen auf die Qualitätskennzahlen (Wierzbicki, Makowski und Wessels, 2000; Turban, Sharda und Delen, 2011).
Viele EUS bieten z.B. die Möglichkeit einer Gewichtung von Optimierungszielen. Darüber hinaus könnte es ein EUS für die Flottenplanung z.B. ermöglichen, die Anzahl der Einsatzfahrzeuge schrittweise zu reduzieren, um die Auswirkungen von Fahrzeugausfällen oder Erkrankungen von Fahrern auf das Betriebsergebnis beobachten zu können. Ein EUS für die Produktionsplanung könnte es z.B. ermöglichen, die Bestellmenge für ein Produkt zu erhöhen oder die Menge verfügbaren Materials zu reduzieren, um eine kurzfristige Erhöhung der Nachfrage oder den Ausfall einer Materiallieferung zu simulieren. Dieses Vorgehen wird auch als Was-Wäre-Wenn-Analyse bezeichnet (Turban, Sharda und Delen, 2011). Beispiele für EUS in unterschiedlichen Anwendungsbereichen sind zu finden bei Igbaria u. a. (1996), Gayialis und Tatsiopoulos (2004), Ruiz, Maroto und Alcaraz (2004) (Flottenplanung), McKay und Wiers (2002) (Produktionsplanung), Fagerholt (2004) (Schiffsflottenplanung).

Modellbasierte EUS für die Planung mit interaktiver Optimierung

Die Konzepte für modellbasierte EUS wurden für die Planung weiterentwickelt, um die Anforderungen einer kooperativen Funktionsaufteilung zu erfüllen. Burstein u. a. (1996) präsentierten eine Sammlung von Diskussionsergebnissen über

ein (aus damaliger Sicht) neues Paradigma für Planungssysteme, der *gemischten Initiative* (engl. mixed initiative). Mensch und Computer werden dabei als gleichberechtigte Agenten betrachtet, die gemeinsam ein bestimmtes Planungsziel erreichen wollen und dazu in jeder Stufe des Lösungsprozesses ihr jeweiliges Wissen auf eigene Initiative einbringen können.
Ähnliche Arbeiten greifen das Prinzip der kooperativen Funktionsaufteilung für Planungssysteme und andere Informationssysteme unter dem Begriff der *interaktiven Optimierung* auf. Da Grob- und Feinplanungsprobleme zur Klasse der Optimierungsprobleme gehören, wird dieser Begriff im Folgenden unter Einbeziehung der Ideen der gemischten Initiative bevorzugt.
Im Vergleich zu herkömmlichen modellbasierten EUS wird der Experte bei der interaktiven Optimierung stärker in die Erstellung einer Alternativlösung einbezogen. Seine Möglichkeiten, unstrukturiertes Wissen auszudrücken, werden dadurch erweitert. Außerdem wird ihm eine bessere Bewertung der Alternativlösung ermöglicht, da er durch seine Beteiligung einen größeren Einblick in die Zusammenhänge des Modells erhält. Die Beteiligung kann auf verschiedene Arten umgesetzt werden:

1. die Einbeziehung des Anwenders in den Prozess der automatischen Lösungserstellung,

2. eine stärkere Beteiligung des Anwenders an der Modell- und Wissensspezifikation oder

3. die teilweise Übernahme des Lösungserstellung durch den Anwender.

Alle Formen der Beteiligung setzen voraus, dass der Anwender an der Benutzerschnittstelle Aktionen zur Beeinflussung der Lösung, der Lösungsverfahren und des Modells ausführen kann. Jede Aktion muss vom Computer sowohl unterstützt (z.B. um fehlerhafte Aktionen zu verhindern) als auch verarbeitet werden (durch Berücksichtigung der Aktion bei der automatischen Lösungserstellung). Unter dem Begriff *Interaktionswerkzeug* wird im Folgenden die Kombination aus einer an der Benutzerschnittstelle angebotenen Aktion und der zu dieser Aktion gehörigen Computerunterstützung bzw. -verarbeitung zusammengefasst (vgl. Abbildung 1.7).
Die interaktive Optimierung wird häufig in Form eines *iterativen Lösungsprozesses* umgesetzt. Dies beruht auf der Erfahrung, dass Anwender ihre Präferenzen für bestimmte Lösungseigenschaften häufig nicht ad hoc, sondern erst anhand eines

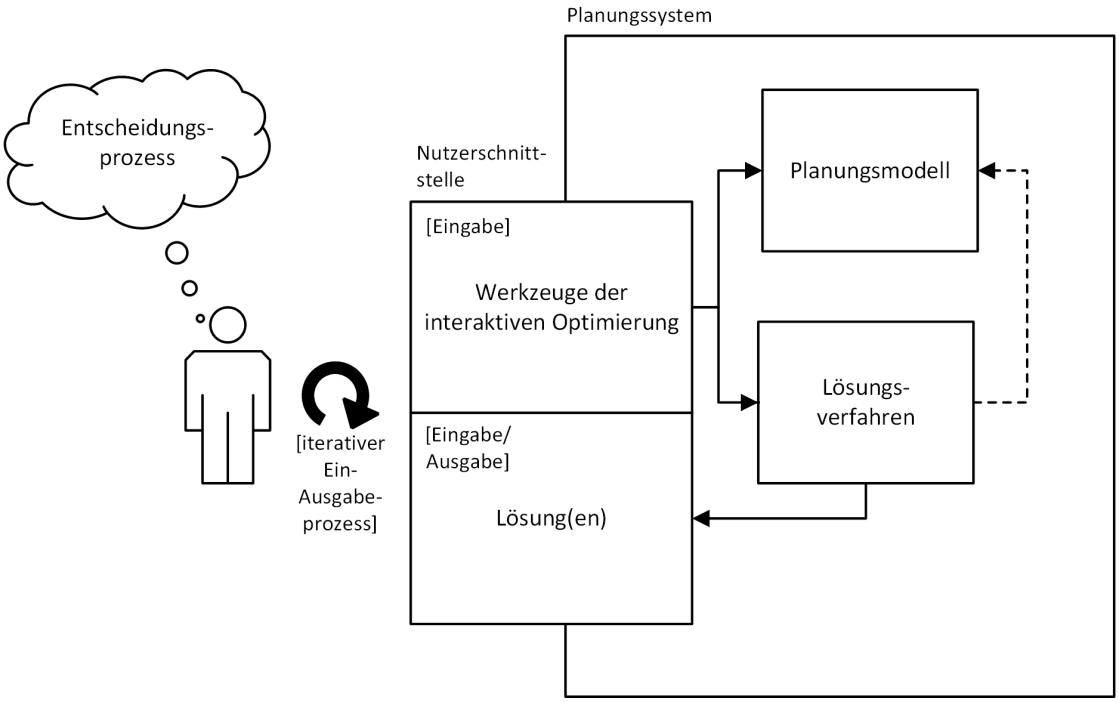


Abbildung 1.7: Modell der interaktiven und iterativen Optimierung

vorgeschlagenen (partiellen) Plans entwickeln können (Gasser, Fischer und Wäfler, 2011). Das System bietet in jedem Iterationsschritt eine (möglicherweise unvollständige) Zwischenlösung an. Sie ermöglicht es dem Anwender, seine Entscheidungen zu bewerten und ggf. Aktionen für den nächsten Iterationsschritt abzuleiten, die mit Hilfe bestimmter Interaktionswerkzeuge umgesetzt werden können. Mögliche Werkzeuge nach dem Vorbild der Punkte 1 bis 3 sind z.B.:

1. Der Anwender kann festlegen, wie die aktuelle Lösung weiter verbessert werden kann, d.h. er kann die Suche nach der optimalen Lösung in eine bestimmte Richtung lenken. Dieses Werkzeug wurde von Klau u. a. (2010) unter der Bezeichnung „Human Guided Search“ (Flotten- und Produktionsplanung, vgl. auch Markus u. a. (2005) und Lesh u. a. (2003)) sowie von Kopfer und Schonberger (2002) (Flottenplanung) und von Müller (2003) (Universitätsstundenplanung, vgl. auch (Müller und Barták, 2002)) vorgestellt.

2. Der Anwender kann das Modell bzw. die Wissensbasis um zusätzliche Bedingungen verfeinern, die bei der Erstellung der nächsten Zwischenlösung beachtet werden müssen. Möglichkeiten zur Modelländerung werden von Nascimento und Eades (2005) sowie von Pu und Faltings (2002) zusammenge-

fasst (vgl. auch Cegarra und Wezel (2011b)). Anwendungsbeispiele sind die Systeme COMIREM (Interaktive Ressourceneinsatzplanung, Smith, Hildum und Crimm (2005)), PTIME (Assistent für die Terminplanung, Berry u. a. (2005)) und MetroNG (Interaktive Stundenplanung an der Universität, Bednárek u. a. (2010)).

3. Der Anwender kann Teile der Lösung manuell modifizieren oder ergänzen. Er kann entscheiden, ob er im nächsten Iterationsschritt mit der manuellen Planung fortfährt oder ob die Lösung (oder ein Teil davon) vom Computer verbessert bzw. vervollständigt werden soll. Manche Systeme erlauben nur eine nachträgliche manuelle Modifikation der automatisch erstellten Lösung, wobei die Änderungen bei einem erneuten Aufruf des Lösungsverfahrens verworfen werden. Im Sinne der gemischten Initiative sollte der Computer den jeweiligen Handlungsspielraum zur manuellen Bearbeitung der Zwischenlösung durch Constraint-Propagierung ermitteln (Smith, Hildum und Crimm, 2005; Pu und Faltings, 2002).

Die Interaktion zwischen Mensch und Computer erfolgt abwechselnd, bis eine vollständige Alternativlösung mit den gewünschten Eigenschaften vorliegt.

1.3.3 Aktuelle Umsetzung der interaktiven Optimierung in industriellen Planungssystemen

Viele der am Markt etablierten Planungssysteme, wie z.B. SAP APO, APplus, Preactor oder Asprova APS (vgl. Abschnitt 1.1.4) lassen sich der Kategorie modellbasierter EUS zuordnen. Sie bieten in der Regel eine Reihe von Funktionen, die eine interaktive Optimierung ermöglichen. Aus dem „Aachener Marktspiegel Business Software (ERP/PPS) 2013/2014“ (Brandenburg u. a., 2013) können die am weitesten verbreiteten Interaktionsmöglichkeiten für den Disponenten entnommen werden:

manuelle Modellanpassung: Die im Planungsmodell hinterlegten Bedingungen können für einzelne Aufgaben oder Ressourcen überschrieben werden. Dazu gehört z.B. die Anpassung der Ausführungsdauer einer Aufgabe, der Rüst- oder Übergangszeiten zwischen einzelnen Aufgaben oder die Anpassung der Kapazität einzelner Ressourcen.

manuelle Planungsentscheidungen: Es können Festlegungen getroffen werden, die bei der automatischen Planung berücksichtigt werden. Dazu gehört z.B. das Zusammenfassen oder Splitten bestimmter Aufgaben, um das Planungsergebnis zu verbessern (z.B. bei der Losgrößenplanung), die manuelle Zuordnung von Ressourcen zu konkreten Aufgaben oder die manuelle zeitliche Verschiebung von Aufgaben. Abbildung 1.8 zeigt eine typische grafische Plantafel, die das manuelle Verschieben von Aufgaben ermöglicht.

Überwachung manueller Planungsentscheidungen: Zuweisungen, die die Randbedingungen des Modells verletzen, werden in der Regel vom System nicht akzeptiert. So ist es z.B. in der Regel nicht möglich, auf eine Ressource an einem Zeitpunkt zuzugreifen, an dem sie nicht verfügbar ist (z.B. aufgrund von Wartungsarbeiten bei Maschinen oder Schichtzeiten beim Personal). Um Fehler von vornherein zu verhindern, wird vom System häufig der Handlungsspielraum für die Umplanung einer Aufgabe visualisiert.

partielle Neuplanung: Um ein unbefriedigendes Planungsergebnis zu verbessern, kann häufig eine vom System unterstützte Umplanung ausgeführt werden. Der Disponent kann dabei wählen zwischen einer Neuplanung einzelner Aufgaben, einer Gruppe von terminlich miteinander verknüpften Aufgaben oder einer Neuplanung von Aufgaben innerhalb eines bestimmten zeitlichen Fensters. Eine Begrenzung auf bestimmte Ressourcen- oder Ressourcengruppen ist häufig ebenfalls möglich.

Beeinflussung automatischer Planungsentscheidungen: Die automatische Planung kann vom Disponenten konfiguriert werden. Es stehen häufig mehrere Heuristiken für die Reihenfolgebildung und Ressourcenzuordnung der einzuplanenden Aufgaben zur Auswahl. Außerdem wird häufig eine Auswahl und Gewichtung von Optimierungszielen ermöglicht.

Simulation: Automatische Planungsdurchläufe können im Rahmen einer Was-Wäre-Wenn-Analyse beliebig oft wiederholt werden. Bei jedem Durchlauf werden bestimmte Zwischenfälle (z.B. Engpässe in der Ressourcenkapazität) oder die Umsetzung bestimmter Planungsentscheidungen (z.B. die Erzwingung bestimmter Fertigstellungstermine) simuliert. Aus den Auswirkungen auf das Planungsergebnis kann der Disponent Rückschlüsse für die Bestimmung der tatsächlichen Planungsentscheidungen ziehen.

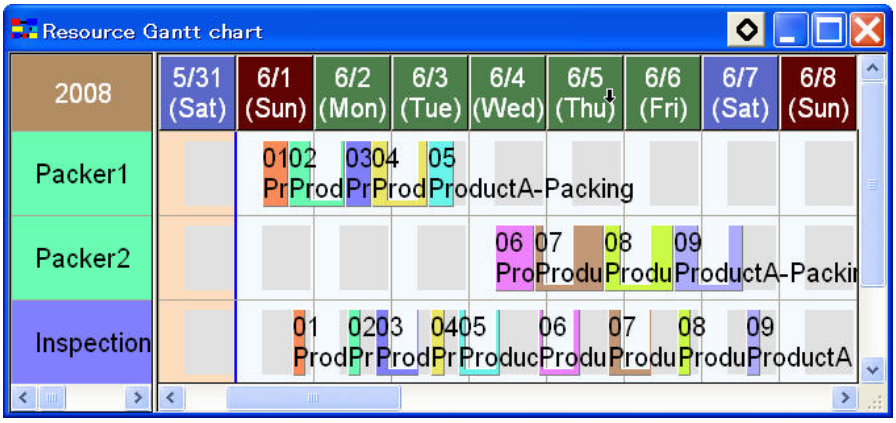


Abbildung 1.8: Grafische Visualisierung eines Produktionsplans als Gantt Chart in Asprova APS. Zusammenhängende Operationen sind durch die gleiche Farbe gekennzeichnet (Asprova, 2013).

1.3.4 Diskussion: Eignung der interaktiven Optimierung für kombinatorische Probleme

Es existieren Konzepte der Funktionsaufteilung, mit denen die optimale Einbeziehung und Unterstützung des Disponenten sichergestellt werden soll. Trotzdem sind einige Experten der Meinung, dass viele Systeme dem Unterstützungsbedarf der Disponenten noch nicht ausreichend gerecht werden (vgl. Abschnitt 1.2.2). Als Ursache kommt, wie im Folgenden begründet wird, eine unzureichende Berücksichtigung der Besonderheiten von kombinatorischen Optimierungsproblemen in Frage.

Kombinatorische Planungsprobleme

Pläne setzen sich aus einer Kombination von zahlreichen Einzelentscheidungen zusammen. Ein Beispiel für eine Einzelentscheidung ist die Festlegung eines Zeitpunktes oder einer Ressource zur Ausführung einer bestimmten Produktionsaufgabe. Planungsprobleme gehören aus diesem Grund zur Klasse der *kombinatorischen Optimierungsprobleme* (Hofstedt und Wolf, 2007; Korte, Vygen und Randow, 2012). Sie sind häufig dadurch gekennzeichnet, dass die Anzahl an Kombinationsmöglichkeiten exponentiell mit der Anzahl möglicher Entscheidungen wächst. Daraus resultiert eine große Menge möglicher Pläne, die als *Suchraum* bezeichnet wird (Hofstedt und Wolf, 2007).

Beispiel 1.6: Ein Disponent möchte Produkte an 20 verschiedene Standorte ausliefern (vgl. Abbildung 1.2). Der Suchraum ist unüberschaubar groß, da es 20! = 2.432.902.008.176.640.000 verschiedene Touren gibt (Problem des Handlungsreisenden, vgl. Görz, Rollinger und Schneeberger (2003)).

Bei kombinatorischen Problemen steht in der Regel für jede Einzelentscheidung nur ein begrenzter Handlungsspielraum zur Verfügung. Zudem können Abhängigkeiten zwischen mehreren Einzelentscheidungen existieren. Diese Einschränkungen werden in der Regel im Modell vorgeschrieben (Abschnitt 1.1.4). Sie führen dazu, dass nicht jede Kombination von Einzelentscheidungen eine Lösung darstellt. Stattdessen liegt im Suchraum eine (häufig ebenfalls sehr große) Teilmenge von Kombinationen als *Lösungsraum* vor (Hofstedt und Wolf, 2007). Die Suche nach einer Lösung, die bestimmte Eigenschaften erfüllt und/oder hinsichtlich bestimmter Ziele optimal ist, gestaltet sich aufgrund der Größe von Such- und Lösungsraum für Mensch und Computer sehr schwierig (Korte, Vygen und Randow, 2012).

Besonderheiten der interaktiven Optimierung bei kombinatorischen Planungsproblemen

Die vom Disponenten gewünschten Lösungseigenschaften können häufig durch eine Kombination von Einzelentscheidungen für bestimmte Aufgaben beschrieben werden.

Beispiel 1.7: Ein Disponent möchte einen Tourenplan so abändern, dass die Lieferung 5 am Ort B vor 10 Uhr und die Lieferung 6 am Ort B nach 16 Uhr stattfindet.

Die gewünschten Eigenschaften können mit den Werkzeugen der interaktiven Optimierung oder durch eine manuelle Modifikation des Plans umgesetzt werden (Abbildung 1.7). Da es sich um operative Planungsentscheidungen handelt, dürfen die im Planungsmodell vorgegebenen Rahmenbedingungen dabei in der Regel nicht verletzt werden (Abschnitt 1.1.4). Dies bedeutet z.B. für die Modellspezifikation, dass keine Bedingungen aus dem Modell entfernt werden dürfen. Es dürfen lediglich Bedingungen hinzugefügt werden, die dazu führen, dass die Lösungsmenge

des geänderten Modells eine Teilmenge der Lösungsmenge des Ausgangsmodells ist. Der Lösungsraum darf also nach Lösungen mit bestimmten Eigenschaften gefiltert werden, aus denen der Computer wiederum die Lösung wählen kann, die bestimmte formal spezifizierte Ziele am besten erfüllt.

Beispiel 1.8: Ein Disponent ist für die Flottenplanung im Unternehmen zuständig. Für eine gegebene Menge an Lieferaufträgen wird die Menge gültiger Pläne beschränkt durch die Anzahl verfügbarer Fahrzeuge, die Arbeitszeiten und Urlaubspläne der Fahrzeugführer und die Regeln zur Zuordnung von Aufträgen zu Fahrzeugtypen (z.B. Transport von Giftstoffen nur in entsprechend gesicherten Fahrzeugen). Auf diese Rahmenbedingungen hat der Disponent keinen Einfluss.

Die bestehenden Interaktionsmodelle sehen jedoch für den Disponenten keinen direkten Zugriff auf den Lösungsraum vor. Er kann versuchen, an der Benutzerschnittstelle Eigenschaften einzugeben, die keinen offensichtlichen Widerspruch mit den bekannten Bedingungen des Ausgangsmodells erzeugen (manuelle Modelländerung oder manuelle Planungsentscheidungen in Abschnitt 1.3.3). Trotzdem kann er dabei aufgrund der Komplexität kombinatorischer Planungsprobleme nicht vorhersehen, ob die Lösung mit der gewünschten Kombination von Eigenschaften im Lösungsraum existiert. So kann es vorkommen, dass seine Modelländerungen vom Computer abgelehnt werden. In diesem Fall muss der Disponent versuchen, die Eigenschaften geringfügig abzuwandeln, sodass sie seine Präferenzen trotzdem noch ausreichend repräsentieren. Die Interaktion zwischen Mensch und Computer könnte dabei wie folgt ablaufen:

1. Der Disponent gibt alle gewünschten Eigenschaften als zusätzliche Bedingungen an der Benutzerschnittstelle ein.
2. Der Computer teilt dem Disponenten mit, dass keine Lösung mit diesen Eigenschaften existiert.
3. Der Disponent modifiziert die zusätzlichen Bedingungen.
4. Der Computer teilt dem Disponenten erneut mit, dass keine Lösung existiert.
5. Der Disponent verzichtet nach einigen Iterationen auf einige gewünschte Eigenschaften. Er reduziert die Bedingungen, um schnell eine Lösung zu erhal-

ten.

Das Szenario zeigt eine Problematik, die bei den vorgestellten Interaktionskonzepten nicht berücksichtigt wird. Der Disponent möchte einerseits zusätzliches Wissen in die Planung einbringen, weiß aber andererseits nicht, welche Kombinationen von Eigenschaften zulässig sind. Die Überwachung manueller Planungsentscheidungen durch den Computer (vgl. Abschnitt 1.3.3) schafft keine Abhilfe: Die Unterstützung erfolgt nur für die aktuelle Einzelentscheidung bzw. für die gerade betrachtete Aufgabe. Nach mehreren gültigen Einzelentscheidungen gelangt der Disponent u.U. an einen Punkt, an dem für die nächste Einzelentscheidung kein oder nicht mehr der gewünschte Handlungsspielraum zur Verfügung steht (eine sogenannte „Sackgasse“). Um den Handlungsspielraum zu erweitern, müssen ggf. bisher getroffene Entscheidungen abgewandelt werden. Für diesen Fall ist keine Computerunterstützung vorgesehen.
Eine indirekte Beteiligung des Disponenten durch partielle Neuplanung oder Konfiguration von Lösungsverfahren (vgl. Abschnitt 1.3.3) bietet dagegen häufig nicht genügend Einflussmöglichkeiten auf konkrete Lösungseigenschaften.
Aufgrund dieser Defizite kann sich die iterative Planung als Prozess von Versuch und Irrtum gestalten, bei dem der Anwender blind im Suchraum navigiert. Dies beeinträchtigt den gesamten Entscheidungsprozess: Wenn die Entwicklung einer einzelnen Lösung nach den Vorstellungen des Disponenten sehr aufwändig ist, können weniger Alternativlösungen einander gegenübergestellt werden. Dies führt dazu, dass die Planungsentscheidungen des Disponenten u.U. auf einer eingeschränkten Sicht auf die betrieblichen Zusammenhänge beruhen. Es wird deutlich, dass Interaktionsmöglichkeiten benötigt werden, die eine effiziente Navigation durch den Lösungsraum ermöglichen.

Gemeinsamkeiten mit Konfigurationsproblemen

Ein Konfigurationsproblem liegt vor, wenn der Anwender ein Produkt durch die Angabe einer Kombination von Merkmalen aus einer Datenbank auswählen soll (Schneeweiss und Hofstedt, 2013).

Beispiel 1.9: In einem Online-Shop können sich Kunden die Merkmale eines T-Shirts vor dem Kauf selbst zusammenstellen. Es werden verschiedene Farben (rot,

schwarz, weiß), Größen (klein, mittel, groß) und Aufdrucke („Men in Black“, „Rettet die Wale!“) angeboten, die miteinander kombiniert werden können. Bestimmte Kombinationsmöglichkeiten, wie z.B. ein weißes T-Shirt mit dem Aufdruck „Men in Black“, sind nicht erlaubt (Fehn, 2014).

Weitere Anwendungsbereiche für Konfigurationsprobleme sind bei Stormer, 2007 zu finden. Kombinatorische Planungsprobleme besitzen im sozio-technischen Kontext viele Gemeinsamkeiten mit Konfigurationsproblemen. Zum einen existieren in beiden Fällen typischerweise Bedingungen, welche die Kombinationsmöglichkeiten der Merkmale einschränken. Zum anderen ist der Lösungsraum, d.h. die Menge gültiger Kombinationen, in beiden Fällen häufig so groß, dass ihre explizite Aufzählung an der Benutzerschnittstelle nicht in Frage kommt (Burstein u. a., 1996; Madsen, 2003). Bei einer interaktiven Auswahl von Merkmalen besteht daher die Gefahr, dass die vom Anwender angegebene Kombination nicht im Lösungsraum enthalten ist. Für Konfigurationsprobleme und für allgemeine kombinatorische Probleme mit ähnlichen Eigenschaften wurden daher Interaktionsrichtlinien entwickelt, die eine effiziente interaktive Merkmalsauswahl ermöglichen (Madsen, 2003; Frayman, 2001).
Es sollte geprüft werden, ob diese Richtlinien auf kombinatorische Planungsprobleme als Teilmenge allgemeiner kombinatorischer Probleme angewendet bzw. für diese spezialisiert werden können.

1.4 Gestaltung der Funktionsaufteilung in Feinplanungssystemen (Forschungsfrage)

1.4.1 Forschungsfrage

Wissenschaftler sind der Meinung, dass die bestehenden Konzepte zur Gestaltung von Planungssystemen noch nicht ausreichend sind, um eine hohe wahrgenommene Nützlichkeit abzusichern (Abschnitt 1.2.2). Insbesondere muss die Entscheidungsunterstützung des Disponenten durch das Planungssystem verbessert werden. Bestehende Konzepte der Funktionsaufteilung vernachlässigen die Tatsache,

dass seine Entscheidungsfreiheit bei kombinatorischen Planungsproblemen durch den Lösungsraum des Planungsmodells beschränkt wird. Ein verbessertes Konzept muss die Auswahl von Plänen mit einer bestimmten Kombination von Eigenschaften einzelner Aufgaben unterstützen. Damit liegen ähnliche Anforderungen vor wie bei Konfigurationsproblemen.
Die Vorberechnung des Lösungsraums durch das Planungssystem ist die theoretisch ideale Unterstützungsmaßnahme. Sie kann jedoch aufgrund der kombinatorischen Komplexität von Planungsproblemen nicht praktisch umgesetzt werden (vgl. Beispiel 1.6 und 2.3). Stattdessen muss ein Kompromiss gefunden werden: Es werden Werkzeuge der interaktiven Optimierung benötigt, die den Aufwand des Entscheidungsprozesses zur Entwicklung und Bewertung von Alternativlösungen so weit wie möglich reduzieren. Folgende Anforderungen gelten:

- Das Auftreten von Sackgassen muss so weit wie möglich reduziert werden. Es sollte weitestgehend verhindert werden, dass der Disponent nicht lösbare Modellspezifikationen formuliert.

- Wenn Sackgassen bzw. nicht lösbare Probleme auftreten (was sich aufgrund der Komplexität nicht ausschließen lässt), müssen die Werkzeuge dem Disponenten eine schnelle und einfache Abwandlung der Planungsentscheidungen ermöglichen, die zur Lösbarkeit des Problems führt.

Als Richtlinie zur Gestaltung von Planungssystemen wird ein minimaler Satz von Interaktionswerkzeugen benötigt, mit dem eine ausreichende und (hinsichtlich der kombinatorischen Komplexität) effiziente Einbeziehung des Disponenten sichergestellt werden kann. Zu diesem Zweck muss geklärt werden, welche der in Abschnitt 1.3.2 vorgestellten Werkzeuge ggf. wiederverwendet oder abgewandelt werden können und welche zusätzlichen Werkzeuge benötigt werden. Die dabei zugrunde liegende Forschungsfrage lautet:

> **Forschungsfrage:** Welches Konzept der Funktionsaufteilung zwischen Mensch und Computer ist bei kombinatorischen Planungsproblemen dafür geeignet, den Disponenten bei der Auswahl eines Plans mit einer bestimmten Kombination von Eigenschaften zu unterstützen?

Mit der Lösung der Forschungsfrage werden die herkömmlichen Konzepte der Funktionsaufteilung (Abschnitt 1.3.2) so ergänzt, dass der Disponent mit Hilfe des Computers schneller und einfacher den gewünschten Plan ermitteln kann. In-

teraktionskonzepte, die für kombinatorische Probleme, wie z.B. Konfigurationsprobleme, entwickelt wurden, werden für Planungsprobleme spezialisiert und ergänzt. Die Arbeit trägt damit dazu bei, die wahrgenommene Nützlichkeit von Planungssystemen zu steigern (vgl. Abschnitt 1.2.2). Diese ist die Voraussetzung für eine erfolgreiche Planung im sozio-technischen System, die sicherstellt, dass die Unternehmensziele, wie z.B. Gewinnsteigerung, Kostensenkung und Zufriedenheit der Mitarbeiter, erreicht werden können. Eine hohe Nutzerakzeptanz bewirkt zudem, dass sich der Einsatz einer Planungssoftware wirtschaftlich rentiert.

Eingrenzung des Forschungsbereiches

Der Forschungsbereich soll folgendermaßen eingegrenzt werden:

a) Es werden modellbasierte Feinplanungssysteme betrachtet, mit denen die detaillierte Zuordnung von Aufgaben zu Ressourcen geregelt wird. Beispiele sind die Zuordnung von

- Transportgüter zu Fahrern,
- Schichten zu Personal,
- Fertigungsaufträgen zu Maschinen.

Die betrachteten Planungsprobleme besitzen typischerweise Randbedingungen, die die Zuordnung von Aufgaben zu Ressourcen oder die zeitliche Anordnung von Aufgaben einschränken. Eine formale Spezifikation der betrachteten Klasse von Planungsproblemen erfolgt in Abschnitt 2.2.

b) Im Vordergrund steht die Funktionsaufteilung zwischen Mensch und Computer bei der interaktiven Erstellung von Plänen. Die Visualisierung von Planungsergebnissen ist nicht Gegenstand dieser Arbeit. Forschungsergebnisse sind auf diesem Gebiet u.a. bei Lambeck u. a. (2012b), Cheng u. a. (2002) oder Luz und Masoodian (2010) zu finden. Diese Arbeiten befassen sich mit neuartigen Darstellungsmöglichkeiten des Lösungsraums, die über das klassische Gantt-Chart hinausgehen.

c) Das Konzept der Funktionsaufteilung ist unabhängig von einer konkreten grafischen Bedienoberfläche. Es kann für tabellenorientierte Eingabeschnittstellen (z.B.

Microsoft Excel) ebenso angewendet werden wie für klassische WIMP-Oberflächen[16]. Sie sind auch geeignet für neuartige Bedienkonzepte, die für berührungsempfindliche Eingabeschnittstellen entwickelt werden (speziell für Planungsprobleme ist dabei auf Lambeck u. a. (2012a) zu verweisen).

Für das entwickelte Konzept kommen u.a. folgende Einsatzbereiche in Frage:

- Feinplanungssysteme für die Produktion
- Stundenplanungssysteme an Universitäten
- Schichtplanungssysteme in Krankenhäusern
- Systeme zur Flotteneinsatzplanung

Abgrenzung zur Masterarbeit

In der vorhergehenden Masterarbeit (Prenzel, 2011) habe ich mich bereits mit einem speziellen Bereich der Thematik dieser Arbeit auseinandergesetzt. Unter Anwendung der Grundlagen von Entscheidungsunterstützungssystemen (EUS) habe ich eine Benutzerschnittstelle für den Anwendungsfall der Routenplanung konzipiert und implementiert. Ein allgemeingültiges Interaktionskonzept, welches auf der in Abschnitt 1.4.1 genannten Zielstellung basiert, lag dabei nicht zugrunde. Die Fallstudie zur Flottenplanung in dieser Arbeit basiert auf dem gleichen Planungsproblem und die dafür neu konzipierte und neu implementierte Benutzeroberfläche enthält einige Interaktionselemente, die bereits in der Masterarbeit verwendet wurden.

1.4.2 Vorgehensweise zur Lösung der Forschungsfrage und Ausblick auf die Kapitel

Die vorliegende Arbeit bewegt sich innerhalb der in Abbildung 1.5 grau hinterlegten Disziplinen. Zur Lösung der Forschungsfrage werden die wissenschaftlichen Erkenntnisse der *Lösung* und *Modellierung* von Planungsproblemen, die Erkenntnisse

[16] WIMP = **W**indows, **I**cons, **M**enus, **P**ointer = Fenster, Symbole, Menüs, (Maus-)zeiger

der *kognitiven Psychologie*, insbesondere der Modellierung von Entscheidungsprozessen, und die Erkenntnisse der *Mensch-Computer-Interaktion* benötigt.

- Aus der formalen Betrachtung von *Planungsmodellen* und Verfahren zur Verarbeitung von Planungsmodellen soll eine allgemeine Problembeschreibung für kombinatorische Planungsprobleme resultieren (Abschnitte 2.2 und 2.3). Darauf beziehen sich sowohl die nachfolgende Beschreibung von Entscheidungsprozessen als auch die Konzepte der Funktionsaufteilung. Dadurch wird die universelle Anwendbarkeit der Forschungsergebnisse innerhalb der betrachteten Problemklasse abgesichert.

- Die Erkenntnisse der *kognitiven Psychologie* werden benötigt, um die Entscheidungsprozesse des Disponenten analysieren zu können. Dabei steht die Suche nach einer Kombination von Einzelentscheidungen für die Eigenschaften eines Plans im Vordergrund (Abschnitt 2.4). Es wird gezeigt, dass sich ein Entscheidungsprozess aus Aktivitäten zur Modellverarbeitung zusammensetzt und auf diese Weise formal beschrieben werden kann (Abschnitt 3.1). Aus der Beschreibung können Anforderungen an die Computerunterstützung zur Reduzierung der wahrgenommenen kombinatorischen Komplexität gewonnen werden (Abschnitt 3.3).

- Bestehende Konzepte der *Mensch-Computer-Interaktion* werden ebenfalls zu Rate gezogen, um grundlegende Anforderungen an die Funktionsaufteilung zu ermitteln (Abschnitte 1.2, 1.3 und 3.2).

- Die ermittelten Anforderungen fließen in Richtlinien zur Funktionsaufteilung zwischen Mensch und Computer ein (Abschnitt 3.4). Um eine intelligente Computerunterstützung konzipieren zu können, werden die Erkenntnisse zur Verarbeitung von Planungsmodellen angewendet. Dazu gehört u.a. die automatische Suche nach Lösungen, die die vom Anwender vorgegebenen Eigenschaften besitzen oder das automatische Ermitteln von Vorschlägen für gültige Eigenschaften.

Abbildung 1.9 fasst die Vorgehensweise zusammen. Im Gegensatz zu dem von MacCarthy, Wilson und Crawford (2001) ursprünglich vorgeschlagenen Framework der Forschungsdisziplinen werden die für die Computerunterstützung ausgewählten Lösungsverfahren auch von den Erkenntnissen der kognitiven Psychologie und von allgemeinen Richtlinien der Mensch-Computer-Interaktion beeinflusst. Die Verbin-

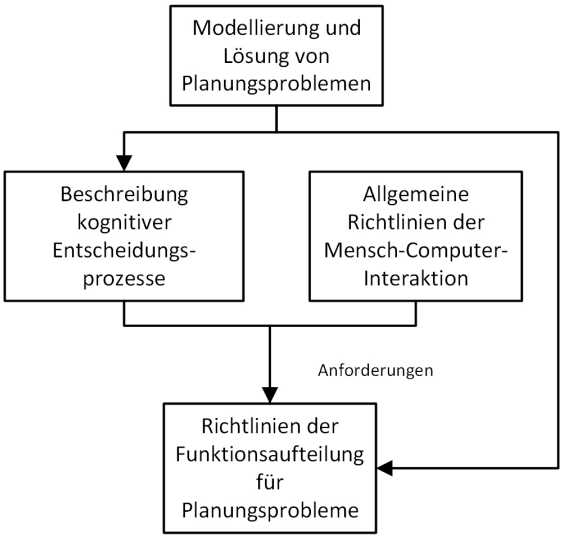


Abbildung 1.9: Herangehensweise zur Lösung der Forschungsfrage

dung wurde in Abbildung 1.5 ergänzt (gestrichelte Linie).
Die Anwendbarkeit der Richtlinien zur Funktionsaufteilung wird in 2 Fallstudien für die Bereiche der Flottenplanung und der Universitäts-Stundenplanung demonstriert. Für die Fallstudie Flottenplanung wird zudem die Vorbereitung und Durchführung eines empirischen Tests geschildert, der im Rahmen dieser Arbeit mit 80 Teilnehmern durchgeführt wurde. Die Testergebnisse weisen nach, dass die vom Anwender gewünschten Pläne effizienter als bisher ermittelt werden können, wenn die Funktionsaufteilung des Planungssystems nach den Richtlinien gestaltet wird.

2 Modellierung und Lösung von praktischen Planungsproblemen

In diesem Kapitel wird der Anwendungsbereich analysiert, auf den sich die in Kapitel 3 entwickelten Konzepte der Funktionsaufteilung beziehen. In Abschnitt 2.1 wird zunächst das Konzept eines Constraint-Erfüllungsproblems (CSP/CSOP) zur formalen Beschreibung von Problemen mit Randbedingungen vorgestellt. Anschließend wird anhand einer Modellstruktur gezeigt, welche Gemeinsamkeiten die Planungsprobleme der Praxis bezüglich ihrer Variablen und Randbedingungen aufweisen. In Abschnitt 2.3 werden verschiedene Verfahren zur Lösung kombinatorischer Planungsprobleme vorgestellt und nach ihrer Komplexität und der angestrebten Lösungsqualität klassifiziert. Im letzten Abschnitt wird der Beitrag des Disponenten zur Lösung von semi-strukturierten Planungsproblemen untersucht, bei denen das zur Lösung erforderliche Wissen nicht vollständig im Modell kodiert ist.

2.1 Allgemeine kombinatorische Constraint-Erfüllungsprobleme (CSP/CSOP)

Ein Constraint-Erfüllungsproblem (engl. **c**onstraint **s**atisfaction **p**roblem = CSP) wird durch ein Tripel (V, D, C) beschrieben (Hofstedt und Wolf, 2007; Dechter, 2003). Dabei ist $V = \{v_1, ..., v_n\}, n = |V|$ eine Menge von Entscheidungsvariablen, D die Menge der zugehörigen Domänen und C die Menge aller Randbedingungen (engl. Constraints) über diesen Variablen. Die Entscheidungsvariablen sind die gesuchten Größen, die nach der Lösung des Problems mit Werten belegt sein müssen. Die Zuordnung von Variablen zu Domänen wird durch die Abbildung $dom : V \rightarrow D$ geregelt. Jeder Variable $v \in V$ dürfen nur Werte zugewiesen werden, die in der zugehörigen Domäne $dom(v)$ enthalten sind.

Die Randbedingungen grenzen den Wertebereich einzelner Variablen (unäre Constraints) oder die gültigen Kombinationen von Werten mehrerer Variablen (n-äre Constraints) ein. Jede Randbedingung $c_i \in C, i \in \{1, ..., |C|\}$ ist ein Paar (S_i, R_i), wobei

- S_i eine Menge von s Variablen ist, über welchen die Randbedingung definiert wird: $S_i = \{v_1, ..., v_s\}, v_j \in V, j \in \{1, ..., s\}, s \in \mathcal{N}^+$ und
- R_i einer Relation entspricht, d.h. einer Teilmenge des Kreuzproduktes der Domänen der beteiligten Variablen: $R_i \subseteq dom(v_1) \times ... \times dom(v_s)$.

Eine *partielle Instanziierung* ist eine Abbildung $\alpha : W \rightarrow \cup D$ mit $W \subseteq V$, sodass für alle $v \in W$ gilt: $\alpha(v) \in dom(v)$. Eine *vollständige Instanziierung* liegt vor, wenn $W = V$. Die Menge aller vollständigen Instanziierungen wird als Suchraum bezeichnet. Eine partielle Instanziierung erfüllt die über sie definierten Randbedingungen, wenn für alle Constraints $(\{v_1, v_2, ..., v_s\}, R) \in C$ mit $v_j \in W, j \in \{1, 2, ..., s\}$ gilt: $(\alpha(v_1), \alpha(v_2), ..., \alpha(v_s)) \in R$. Eine partielle Instanziierung, die ihre Randbedingungen erfüllt, wird als *partielle Lösung* bezeichnet. Eine partielle Lösung mit $W = V$ ist eine *Lösung*[1]. Die Menge L aller Lösungen wird als Lösungsraum bezeichnet.
Aufgrund der Vernetzung von Variablen durch Randbedingungen werden CSPs oft als Constraint-Netzwerke bezeichnet (Dechter, 2003). Ein Constraint-Netzwerk mit binären Randbedingungen lässt sich als Graph darstellen, bei dem die Knoten Variablen und die Kanten zwischen ihnen die Randbedingungen repräsentieren.

Beispiel 2.1: Vier Aufgaben sollen zeitlich auf einer Ressource angeordnet werden. Ihre Startzeiten werden durch die Variablen $s_i, i \in \{1, ..., 4\}$ mit $dom(s_i) \in \{0, 5, 10, 15\}$ repräsentiert. Jede Aufgabe i hat eine Dauer von 5 Zeiteinheiten. Die Reihenfolgeabhängigkeiten zwischen den Aufgaben werden durch Randbedingungen bestimmt:

- Aufgabe 1 soll vor Aufgabe 2 ausgeführt werden: $s_1 + 5 \leq s_2$
- Aufgabe 1 soll vor Aufgabe 3 ausgeführt werden: $s_1 + 5 \leq s_3$
- Aufgabe 2 soll vor Aufgabe 4 ausgeführt werden: $s_2 + 5 \leq s_4$

[1] Anmerkung: Nicht jede partielle Lösung lässt sich zu einer Lösung erweitern (vgl. Abschnitt 2.3.6, k-Konsistenz).

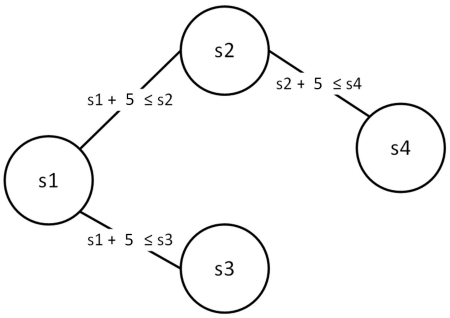


Abbildung 2.1: Netzwerk von Randbedingungen zwischen den Variablen $s_i \in \{0, 5, 10, 15\}$, $i \in \{1, ..., 4\}$

Da die Aufgaben zeitlich überlappungsfrei angeordnet werden sollen, gilt zusätzlich das „serialize“-Constraint (Hofstedt und Wolf, 2007). Abbildung 2.1 zeigt den zugehörigen Constraint-Graph. Aus den Randbedingungen ergeben sich die möglichen Lösungen, wie z.B. $\alpha(s_1) = 0, \alpha(s_2) = 5, \alpha(s_3) = 10, \alpha(s_4) = 15$.

Ein CSP kann mit der Zielsetzung formuliert werden, eine, mehrere oder (im endlichen Fall) alle Lösungen im Lösungsraum L des Problems zu finden. Darüber hinaus wird häufig eine *optimale* Lösung des Problems gesucht. Dabei wird die Problemformulierung durch eine *Leistungskennzahl* oder *Zielfunktion* f ergänzt, die jeder Lösung einen numerischen Wert $k \in \mathcal{R}$ zuordnet $f : L \to \mathcal{R}$ (Dechter, 2003). Es entsteht ein Constraint-Erfüllungs- und Optimierungsproblem (CSOP) (V, D, C, f), bei dem eine Lösung $l \in L$ gesucht wird, die den Zielfunktionswert $f(l)$ je nach Aufgabenstellung minimiert oder maximiert. Die Menge $O_{min}, O_{min} \subseteq L$ der optimalen Lösungen eines CSOPs wird für Minimierungsprobleme definiert als

$$O_{min} = \{l | l \in L \wedge \forall l' \cdot l' \in L \wedge l' \neq l \Rightarrow f(l) \leq f(l')\} \tag{2.1}$$

Analog dazu wird $O_{max}, O_{max} \subseteq L$ für Maximierungsprobleme wie folgt definiert:

$$O_{max} = \{l | l \in L \wedge \forall l' \cdot l' \in L \wedge l' \neq l \Rightarrow f(l) \geq f(l')\} \tag{2.2}$$

Wenn mehrere Optimierungsziele existieren, liegt ein Constraint-Erfüllungsproblem und multikriterielles Optimierungsproblem (CSMOP) vor, welches durch das Quadrupel (V, D, C, F) mit einer Liste von m Zielfunktionen $F = \{f_1, ..., f_m\}$ definiert ist (Jaâfar, Khayati und Ghédira, 2004). Bei der Lösung des CSMOPs können mehrere, sogenannte *pareto-optimale* Lösungen ermittelt werden (Ehrgott, 2005).

Eine Lösung ist pareto-optimal, wenn sie nicht mehr hinsichtlich einer Zielfunktion verbessert werden kann, ohne dass sie sich hinsichtlich einer anderen Zielfunktion verschlechtert. Die Menge aller pareto-optimalen Lösungen wird als *Pareto-Front* bezeichnet. Die Auswahl der am besten geeigneten Lösung wird dem menschlichen Entscheider überlassen (Bui u. a., 2012). Alternativ kann eine gemeinsame Zielfunktion $f_{pareto} : L \rightarrow \mathcal{R}$ definiert werden, die als gewichtete Summe der ursprünglichen Zielfunktionen ausgedrückt wird. Seien $w_1, ..., w_m$ die Gewichte, mit denen $f_1, ..., f_m$ jeweils in die Bewertung eingehen, so ergibt sich für die neue Zielfunktion:

$$f_{pareto}(l) := \sum_{i \in \{1,...,m\}} w_i * f_i(l), l \in L \tag{2.3}$$

Analog zu den Gleichungen 2.1 bzw. 2.2 wird eine Lösung $l \in L$ gesucht wird, die den Zielfunktionswert $f_{pareto}(l)$ minimiert bzw. maximiert. Ansätze und Verfahren der multikriteriellen Optimierung werden in (Ehrgott, 2005) behandelt. In dieser Arbeit wird die Abkürzung CSOP stellvertretend für CSOPs und CSMOPs verwendet.

Im Bereich der Planung spielen *kombinatorische* CSPs/CSOPs eine besondere Rolle (Korte, Vygen und Randow, 2012; Brucker und Knust, 2012a; Gendreau und Potvin, 2005; Ausiello, 1999). Bei diesen Problemen liegen endliche Domänen mit diskreten Werten vor. Es wird eine Lösung in einem endlichen Lösungsraum L gesucht, der wiederum eine Teilmenge eines endlichen Suchraums $S, L \subseteq S$ ist. Analog dazu wird bei einem *kombinatorischen Optimierungsproblem* in L eine Lösung mit optimalen Zielfunktionswert gesucht. Die Größe des Suchraums wächst exponentiell mit der Anzahl der Variablen $|V|$, d.h. mit $\mathcal{O}(|D|^{|V|})$, wobei $|D|$ die maximale Größe der Domänen des CSPs/CSOPs ist.

In dieser Arbeit werden ausschließlich kombinatorische CSPs/CSOPs betrachtet. Wenn die Begriffe „Planungsproblem“, „CSP“, „CSOP“ verwendet werden, sind kombinatorische Planungsprobleme bzw. kombinatorische CSPs/CSOPs gemeint.

2.2 Eine Modellstruktur für allgemeine kombinatorische Planungsprobleme

Die kombinatorischen Planungsprobleme, die in der Praxis auftreten, sind sehr vielfältig. Trotzdem lassen sich in ihrer Problemstruktur zahlreiche Gemeinsam-

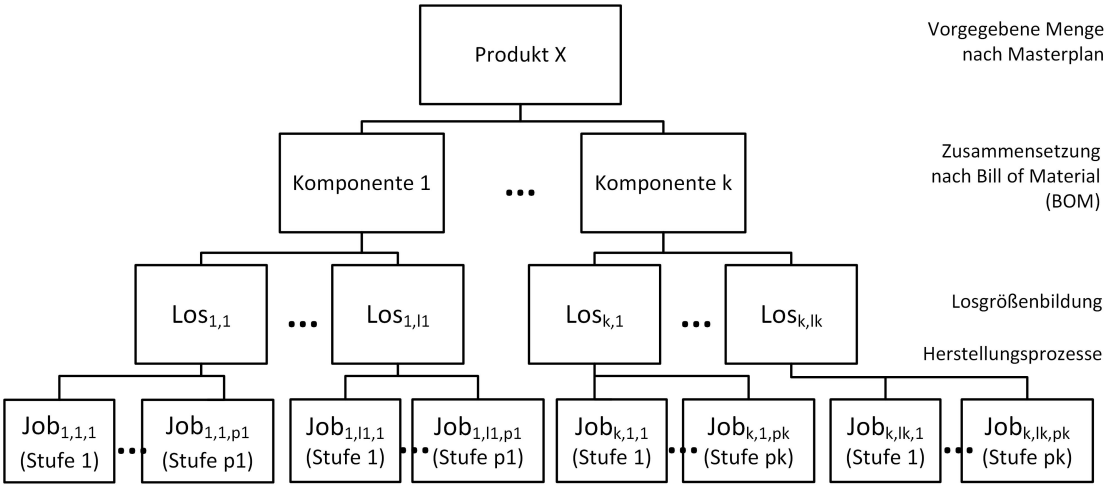


Abbildung 2.2: Aggregationsstufen in einer Produktionsumgebung

keiten finden. Sie werden im Folgenden in einer allgemeingültigen Modellstruktur zusammengefasst, die das Grundgerüst für die in dieser Arbeit betrachteten Probleme bilden soll. Die folgenden Vorüberlegungen dienen dazu, einige grundlegende Anforderungen an die Modellstruktur zu ermitteln.

2.2.1 Vorüberlegungen

Die Lösung von Planungsproblemen dient im Allgemeinen dazu, die folgenden Fragen zu beantworten:

1. Wie viele und welche Aufgaben sollen in einem bestimmten Zeitraum an einem bestimmten Ort ausgeführt werden?
2. In welcher Reihenfolge sollen die Aufgaben ausgeführt werden?
3. Wann sollen die Aufgaben ausgeführt werden?
4. Durch welche Ressourcen sollen die Aufgaben ausgeführt werden?

Frage 1 trifft z.B. auf allgemeine Produktionsplanungsprobleme zu und wird im Rahmen der Grobplanung bzw. Masterplanung beantwortet. Dabei wird ein mittelfristiger Planungshorizont in grobe Zeitabschnitte unterteilt (z.B. in einzelne Wochen). Für jeden Unternehmensstandort wird festgelegt, in welchem Zeitabschnitt welche Produktmenge hergestellt werden muss, sodass die Nachfrage der Kunden

gedeckt wird (Staedler u. a., 2012). Ein formales Modell dieses Problems ist z.B. das in (Pochet, 2001) beschriebene „mehrstufige Losgrößenproblem“ (**m**ulti-**l**evel **l**ot **s**izing **p**roblem = MLLSP).
Aus dem Masterplan können die einzelnen (Produktions-)aufgaben abgeleitet werden. In vielen Anwendungsfällen werden dabei von der Grob- bis zur Feinplanung ein oder mehrere Aggregierungsstufen durchlaufen (Kurbel, 2013). Eine Aufgabe wird als aggregiert betrachtet, wenn sie nicht selbst auf einer bestimmten Ressource durchgeführt wird, sondern in weitere Teil- oder Unteraufgaben zerlegt bzw. disaggregiert werden muss.

Beispiel 2.2: Abbildung 2.2 zeigt ein Beispiel für eine 4-stufige Aggregierung von Aufgaben in allgemeinen Produktionsplanungsproblemen: Jedes Produkt wird aus k verschiedenen Komponenten zusammengebaut. Die Vorschrift liefert dafür eine Stückliste (engl. **b**ill **o**f **m**aterial = BOM), die für den jeweiligen Produkttyp spezifisch ist[2]. In einem Materialbedarfsplan (eng. **m**aterial **r**equirements **p**lan = MRP) wird festgelegt, zu welchem (aggregierten) Zeitpunkt die Komponenten der im Masterplan ausgewiesenen Endprodukte hergestellt werden müssen, sodass diese pünktlich montiert und ausgeliefert werden können. Um die Produktion flexibler zu gestalten, können die Produktionsmengen in Losgrößen unterteilt werden, die jeweils am Stück bearbeitet werden (Kurbel, 2013; Heisig, 2002). Im Beispiel in Abbildung 2.2 wird die Produktion für Komponente $i, i \in \{1, ..., k\}$ jeweils in l_i Lose unterteilt.
Bei der Bearbeitung muss in der Regel ein mehrstufiger Produktionsprozess eingehalten werden. Eine Komponente könnte z.B. die Prozessstufen „Schneiden“, „Schleifen“ und „Lackieren“ erfordern (Staedler u. a., 2012). Im Beispiel in Abbildung 2.2 müssen für jedes Los von Komponente $i, i \in \{1, ..., k\}$ p_i Aufgaben (Jobs) im Feinplan zugeordnet werden, die den p_i Prozessstufen entsprechen.

Während Planungsentscheidungen gemäß Frage 1 regeln, *welche* Aufgaben ausgeführt werden müssen (engl. planning), beschäftigen sich die Planungsentscheidungen gemäß der Fragen 2 bis 4 damit, *wie* die erzeugten Aufgaben ausgeführt werden (engl. scheduling). Letztere können auf jeder Aggregierungsebene getroffen werden (Kovács, 2005). Im Rahmen der Feinplanung werden nur Aufgaben betrachtet, die

[2]Hier wird vereinfachend angenommen, dass die Stückliste nicht selbst hierarchisch unterteilt ist. Andernfalls entstehen zusätzliche Aggregationsstufen.

nicht weiter disaggregiert werden können (Staedler u. a., 2012).
Bei m Ebenen beeinflusst jede Entscheidung für ein Aggregat auf einer Ebene $n, n \in \{1, ..., m\}$ den Planungsspielraum für die damit verknüpften disaggregierten Aufgaben auf der nächst tieferen Ebene $n-1$. Die Entscheidungen über Zeitpunkte und Ressourcen können selbst aggregiert sein. Auf höheren Ebenen ist es z.B. häufig zweckmäßig, Aufträgen zunächst eine bestimmte Ressourcengruppe oder eine Kombination von Ressourcengruppen zuzuweisen, sodass auf der untersten Ebene ein gewisser Handlungsspielraum für die Wahl der konkreten Ressourcen übrig bleibt (Staedler u. a., 2012).
Bei der Erzeugung und bei der Ausführungssteuerung von Aufgaben müssen in der Regel viele *Parameter* berücksichtigt werden, aus denen sich bestimmte Randbedingungen ergeben. In der Produktionsplanung existieren z.B. häufig alternative Herstellungsprozesse. In Beispiel 2.2 könnte z.B. jedem Produkt ein zusätzlicher Parameter zugeordnet werden, der den mit einem bestimmten Herstellungsprozess verbundenen Ausführungsmodus festlegt (Saygin und Kilic, 1999). Typische Parameter für Aufgaben in der Feinplanung sind die Priorität oder die erforderliche Qualifikation der Ressource.

2.2.2 Eine Modellstruktur als Grundgerüst für kombinatorische Planungsprobleme

Eine allgemeine Modellstruktur für Planungsprobleme beschreibt, welche Arten von Entscheidungsvariablen und Parametern auf unterschiedlichen Aggregationsebenen vorliegen und welche Arten von Randbedingungen innerhalb einer Aggregationsebene sowie zwischen unterschiedlichen Aggregationsebenen existieren. Da in dieser Arbeit Feinplanungsprobleme im Vordergrund stehen, liegt der nachfolgenden Modellstruktur die Annahme zugrunde, dass bereits eine Master- bzw. Losgrößenplanung vorgenommen wurde und somit eine Menge von Aggregaten sowie sich daraus ergebenden Feinplanungsaufgaben vorliegt. Es wird definiert, wie sich Planungsentscheidungen auf unterschiedlichen Aggregationsebenen auf den Handlungsspielraum der Feinplanung auswirken. Jedes CSP/CSOP, welches sich auf die Modellstruktur abbilden lässt, gehört zur Klasse der kombinatorischen Planungsprobleme, die in dieser Arbeit betrachtet werden.

Definition von Aufgaben, Ressourcen und Zeitpunkten

Es wird angenommen, dass die Planung für einen begrenzten Planungszeitraum vorgenommen wird, für den eine feststehende Nachfrage an Gütern oder Dienstleistungen vorliegt. Die Nachfrage bestimmt eine Menge von n nicht weiter zerlegbaren Aufgaben $AS = \{a_1, ..., a_n\}$, die im Rahmen der Feinplanung bestimmten Ressourcen und Ausführungszeitpunkten zugeordnet werden müssen.
Es sind u verschiedene Ressourcen $RS = \{r_1, ..., r_u\}$ vorhanden. An der Ausführung einer Aufgabe ist jeweils eine Ressource oder eine Kombination von Ressourcen[3] beteiligt. Es sei RCS die Menge möglicher Ressourcen und Ressourcenkombinationen $RCS \subseteq \mathcal{P}(RS)$. Die Ausführungszeitpunkte der Aufgaben werden in einem endlichen Planungshorizont P mit $h \in \mathcal{N}^+$ diskreten Zeitpunkten gespeichert: $P = \{1, ..., h\}$. Jeder Aufgabe a soll durch die Lösung des Planungsproblems ein möglicher Startzeitpunkt a_{start}, ein Endzeitpunkt a_{end} sowie eine Ressourcenkombination a_{res} zugeordnet werden. Die Domänen der Zeitvariablen sind Teilmengen des Planungshorizontes: $dom(a_{start}) \subseteq P$, $dom(a_{end}) \subseteq P, \forall a \in AS$. Außerdem gilt für alle $a \in AS$: $a_{start} \leq a_{end}$. In dieser Arbeit wird vorausgesetzt, dass die Aufgaben während ihrer Ausführung nicht unterbrochen werden. Die Domäne der Ressourcenvariable schreibt vor, welche Ressourcenkombinationen für die Ausführung in Frage kommen: $dom(a_{res}) \subseteq RCS$.

Verknüpfung von Aufgaben und Aggregaten

Die Aufgaben $a \in AS$ werden über potenziell mehrere Aggregationsebenen in sogenannten Aggregaten zusammengefasst (siehe Abbildung 2.3). Die Anzahl der Ebenen sei $m + 1$. Sei $AG = \mathcal{P}(AS)$ die Menge der Aggregate. Auf jeder Ebene $e, e \in \{0, ..., m\}$ existiert eine Menge AG^e solcher Aggregate, die untereinander disjunkt sind, dabei in Vereinigung aber alle Aufgaben umfassen: $\bigcup_{ag \in AG^e} ag = AS$ und $ag_i \cap ag_j = \emptyset$, $i \neq j$, $\forall i, j \in \{1, ..., |AG^e|\}$. Die Aggregate auf Ebene 0 (Feinplanungsebene) seien einelementige Mengen von Aufgaben, d.h. $AG^0 = \{ag_1^0, ..., ag_n^0\}$ mit $ag_i^0 = \{a_i\}, i \in \{1, ..., n\}$.
Ein Aggregat einer Ebene e vereinigt jeweils Aggregate der darunter liegenden Ebene $e - 1$. Die Funktion $aggregat : AG \rightarrow AG$ liefert für ein Aggregat ag

[3] Es kann z.B. eine Kombination aus Personal mit einer bestimmten Qualifikation, bestimmten Werkzeugen und bestimmten Maschinen erforderlich sein (Asprova, 2013).

Ebene	
2	$ag^2_1 = \{a_1, a_2, a_3, a_4, a_5\}$
1	$ag^1_1 = \{a_1, a_2, a_3\}$ $ag^1_2 = \{a_4, a_5\}$
0	$\{a_1\}$ $\{a_2\}$ $\{a_3\}$ $\{a_4\}$ $\{a_5\}$

Abbildung 2.3: Beispielhafte Zuordnung von Aufgaben zu Aggregaten über 2 Aggregationsebenen mit $AG^0 = \{ag^0_1, ..., ag^0_5\}$, $AG^1 = \{ag^1_1, ag^1_2\}$, $AG^2 = \{ag^2_1\}$. Es gilt $ag^0_i = \{a_i\}, i \in \{1, ..., 5\}$.

der Ebene e das zugeordnete Aggregat der darüber liegenden Ebene $e + 1$, z.B. $aggregat(\{a_1\}) = ag^1_1$ und $aggregat(ag^1_2) = ag^2_1$ (Abbildung 2.3). Es gilt: $ag \subseteq aggregat(ag)$.

Verknüpfung von Zeitzuweisungen zwischen den Aggregaten

Im Rahmen der Masterplanung wird für jedes Aggregat ein Zeitrahmen ermittelt, in welchem die daraus abgeleiteten Aufgaben ausgeführt werden müssen (vgl. Abschnitt 2.2.1). Jedes Aggregat $ag \in AG^e$ auf einer Ebene $e \in \{1, ..., m\}$ sei daher über die Abbildungen $start : AG^e \rightarrow P$ und $end : AG^e \rightarrow P$ mit $start(ag) \leq end(ag)$ einem Ausführungszeitraum zugeordnet. Die Verknüpfung der Ebenen erfolgt über die nachfolgenden Randbedingungen.
1.) Jede Aufgabe $a \in AS$ muss innerhalb des Zeitraumes ausgeführt werden, der für das zugehörige Aggregat $aggregat(\{a\})$ festgelegt wurde:

$$a_{start} \geq start(aggregat(\{a\})), \forall a \in AS \tag{2.4}$$

$$a_{end} \leq end(aggregat(\{a\})), \forall a \in AS \tag{2.5}$$

2.) Für jedes Aggregat $ag^e \in AG^e$ auf Ebene e dürfen nur Start- und Endzeitpunkte vergeben werden, die innerhalb des Zeitraumes liegen, der für das übergeordnete Aggregat $aggregat(ag^e)$ festgelegt wurde:

$$start(ag^e) \geq start(aggregat(ag^e)), \quad \forall ag^e \in AG^e, e \in \{1, ..., (m-1)\} \tag{2.6}$$

$$end(ag^e) \leq end(aggregat(ag^e)), \quad \forall ag^e \in AG^e, e \in \{1, ..., (m-1)\} \tag{2.7}$$

Verknüpfung von Ressourcenzuweisungen zwischen den Aggregaten

Für jedes Aggregat können Festlegungen für die Ressourcenauswahl getroffen werden, die von den Aggregaten tiefer liegender Ebenen berücksichtigt werden müssen. Die Menge gültiger Ressourcenkombinationen kann für ein Aggregat $ag^e, e >= 1$ auf eine Teilmenge von RCS beschränkt werden, um z.B. nur Ressourcen einer bestimmten Kategorie (Standort, Klassifikation, Qualifikation etc.) für die Ausführung der daraus abgeleiteten Aufgaben zuzulassen. Die Festlegung erfolgt auf Ebene $e, e >= 1$ durch die Abbildung $res : AG \rightarrow \mathcal{P}(RCS)$. Für jede Aufgabe $a \in AS$ mit $aggregat(\{a\}) = ag^1$ gilt dann die Randbedingung: $a_{res} \in res(ag^1)$. Außerdem gelte für alle $e, e > 1$ für jedes untergeordnete Aggregat ag^{e-1} mit $aggregat(ag^{e-1}) = ag^e$ die Randbedingung: $res(ag^{e-1}) \subseteq res(ag^e)$.

2.2.3 Anwendung der Modellstruktur auf Problemklassen der Feinplanung

Im Folgenden werden einige Problemklassen der Feinplanung vorgestellt, die der allgemeinen Modellstruktur folgen.

Ressourcenbeschränkte Projektplanung (RCPSP)

Die ressourcenbeschränkte Projektplanung (**r**esource **c**onstraint **p**roject **s**cheduling = RCPSP) ist eine Oberklasse für zahlreiche praktische Feinplanungsprobleme. Die Aufgabenstellung besteht darin, ein Projekt zu planen, welches aus n Aufgaben besteht, zwischen denen Reihenfolgeabhängigkeiten bestehen können (Hartmann und Briskorn, 2010). Dabei beanspruchen die Aufgaben während ihrer Ausführung von jeder Ressource eine bestimmte Kapazität. Da die Gesamtkapazität einer Ressource beschränkt ist, können nicht alle Aufgaben, die sie benötigen, gleichzeitig ausgeführt werden. Daher besteht das Ziel der Projektplanung darin, die Aufgaben zeitlich so auf den Ressourcen anzuordnen, dass die Gesamtdauer D zur Durchführung des Projektes minimal ist (Brucker und Knust, 2012b).

Definitionen gemäß Modellstruktur:

- $AS = \{a_1, ..., a_n\}$

- $RS = \{r_1, ..., r_u\}$, $RCS = \{RS\}$
- $\forall a \in AS : dom(a_{start}) = P, dom(a_{end}) = P$
- keine Aggregationsebenen, d.h. $m = 0$

Weitere Definitionen:
Die von einer Aufgabe $a \in AS$ von der Ressource $r \in RS$ benötigte Kapazität ist durch die Abbildung $cap_{aufgabe} : AS \times RS \to \mathcal{R}$ gegeben. Die von einer Ressource $r \in RS$ bereitgestellte Kapazität ist gegeben durch $cap_{resource} : RS \to \mathcal{R}^+$.
Die Abbildung $dur : AS \to P$ beschreibt die Dauer der Ausführung einer Aufgabe. Mögliche Reihenfolgevorgaben sind in einer Menge E von Indexpaaren enthalten. Wenn $(i, j) \in E$, dann muss Aufgabe a_i vor Aufgabe a_j ausgeführt werden.
Weiterhin wird eine Funktion $active : AS \times P \to \{0, 1\}$ definiert, die angibt, ob eine Aufgabe a zum Zeitpunkt t aktiv ist (Funktionswert 1) oder nicht (Funktionswert 0). Es gilt:

$$active(a, t) = 1 \Leftrightarrow a_{start} \leq t \leq a_{end}, \quad \forall a \in AS, t \in \{1, ..., h\} \tag{2.8}$$

Randbedingung 2.9 stellt die Einhaltung der Ausführungdauer sicher:

$$a_{start} + dur(a) = a_{end}, \quad \forall a \in AS \tag{2.9}$$

Randbedingung 2.10 stellt die Einhaltung der Reihenfolgeabhängigkeiten sicher:

$$a_{i_{start}} + dur(a_i) \leq a_{j_{start}}, \quad \forall (i, j) \in E \tag{2.10}$$

Randbedingung 2.11 verhindert, dass die Kapazität einer Ressource zu einem bestimmten Zeitpunkt überschritten wird:

$$\sum_{a \in S_t} cap_{aufgabe}(a, r) \leq cap_{resource}(r), \quad \forall t \in P, \forall r \in RS \tag{2.11}$$

Dabei enthält $S_t = \{a \in AS | active(a, t) = 1\}$ alle Aufgaben, die sich zum Zeitpunkt t in Bearbeitung finden.
Das Optimierungsziel ist die Minimierung der Gesamtdauer D,

$$\min D = max_{\forall a \in AS}(a_{end}) \tag{2.12}$$

Weitere Modellvarianten und Optimierungsziele des RCPSP sind bei Brucker und Knust (2012b), Artigues, Demassey und Néron (2013) und Kuster, Jannach und Friedrich (2007) zu finden.

Werkstattplanung (JSSP)

Im Bereich der Produktionsplanung wird häufig das Modell der Werkstattplanung (**j**ob **s**hop **s**cheduling = JSSP) eingesetzt. Den Ausgangspunkt bildet ein Materialbedarfsplan und eine Unterteilung von Produktionsmengen in Lose (vgl. Beispiel 2.2). Sobald die Losgrößen feststehen, müssen für jedes Los ein oder mehrere produkt- oder komponentenspezifische Verarbeitungs- oder Montageprozesse auf festgelegten Ressourcentypen eingeplant werden. Zur Modellierung wird jedes Los als Aggregat aufgefasst.

Definitionen gemäß Modellstruktur:

- $AS = \{a_1, ..., a_n\}$
- $RS = \{r_1, ..., r_u\}$, $RCS = \{\{r_1\}, ..., \{r_u\}\}$
- $\forall a \in AS : dom(a_{start}) = P, dom(a_{end}) = P$
- eine Aggregationsebene mit o Aggregaten, d.h. $m = 1$, $AG^1 = \{ag_1^1, ..., ag_o^1\}$.

Weitere Definitionen:
Jedem Aggregat $ag_i^1, i \in \{1, ..., o\}$ werden p_i Aufgaben $a_{ij}, j \in \{1, ..., p_i\}$ mit $aggregat(a_{ij}) = ag_i^1$ zugeordnet, die den einzelnen Verarbeitungsprozessen entsprechen. Für jede Aufgabe a_{ij} wird dabei eine Ressource, d.h. ein Element aus RCS fest vorgegeben. Die Abbildungen dur, $active$, $cap_{aufgabe}$ und $cap_{resource}$ werden analog zum RCPSP definiert. Es gilt die Randbedingung 2.9. Da die Verarbeitungsprozesse aufeinander aufbauen, besteht zwischen den Aufgaben eines Aggregats eine fest vorgegebene produkt- oder komponentenspezifische Reihenfolge:

$$a_{ij} + dur(a_{ij}) \leq a_{i,j+1}, \quad i \in \{1, ..., o\}, j \in \{1, ..., p_{i-1}\} \tag{2.13}$$

Das JSSP sieht vor, dass nicht mehr als eine Aufgabe gleichzeitig von einer Maschine ausgeführt werden kann. Jede Aktivität beansprucht die Maschine also zu 100%. Dies kann über Randbedingung 2.11 mit $cap_{aufgabe}(a, r) = 1, \forall a \in AS, \forall r \in a_{res}$ und $cap_{resource}(r) = 1, \forall r \in RS$ modelliert werden.
Dieses Problem und weitere Varianten des JSSPs sowie weitere Produktionsplanungsmodelle sind bei (Pinedo (2012), Kapitel 7) und Brucker und Knust (2012b) zu finden. Neben der Minimierung der Gesamtdauer (vgl. Gleichung 2.12) werden

zum Teil weitere Zielfunktionen eingesetzt, wie z.B.:

- Minimierung der Summe der Rüstkosten, die beim Einrichten der Maschinen anfallen,
- Minimierung der Summe der Kosten für die verspätete Fertigstellung von Aufträgen.

Flottenplanung (VRPTW)

Wie beim allgemeinen RCPSP gibt es auch bei einem Flottenplanungsproblem (hier: **v**ehicle **r**outing **p**roblem with **t**ime **w**indows = VRPTW) eine Menge AS mit n Aufgaben.

Definitionen gemäß Modellstruktur:

- $AS = \{a_1, ..., a_n\}$
- $RS = \{r_1, ..., r_u\}$, $RCS = \{\{r_1\}, ..., \{r_u\}\}$
- $\forall a \in AS : dom(a_{start}) = P, dom(a_{end}) = P$
- keine Aggregationsebenen, d.h. $m = 0$

Weitere Definitionen:
Die Aufgaben sind jeweils mit einem Zeitfenster, einer Dauer $dur : AS \rightarrow \mathcal{N}^+$ und mit einem geografischen Ort $loc : AS \rightarrow LS$ verknüpft. Die endliche Menge LS definiert alle benötigten Orte. Das Zeitfenster einer Aufgabe wird durch den frühesten Startzeitpunkt $estart : AS \rightarrow P$ und den spätesten Startzeitpunkt $lstart : AS \rightarrow P$ begrenzt. Alle Aufgaben müssen innerhalb ihres Zeitfensters an den jeweiligen Orten ausgeführt werden. Es kann sich z.B. um Besuchstermine eines Außendienstmitarbeiters bei seinen Kunden oder um Liefertermine für Waren handeln. Es stehen u Fahrzeuge zur Verfügung, um innerhalb des Planungszeitraumes alle Orte zu besuchen. Es ist das Ziel der Planung, alle Aufgaben auf die Fahrzeuge zu verteilen, sodass für jedes Fahrzeug Touren mit einer minimalen Länge gebildet werden können.
Jedem Fahrzeug $r_k, k \in \{1, ..., u\}$ sind zwei Pseudo-Aufgaben $a_{k,s}$ und $a_{k,e}$ zugeordnet, die das Depot markieren, an dem die Tour des Fahrzeugs beginnt und

endet.
Es gilt Gleichung 2.9 zur Einhaltung der Ausführungsdauer. Für die Einhaltung der Zeitfenster gilt:

$$a_{start} \geq estart(a_i) \wedge a_{start} \leq lstart(a_i), \quad \forall a \in AS \tag{2.14}$$

Die Entfernungen (z.B. in km) zwischen den Besuchsorten werden in einer Entfernungsmatrix festgehalten. Sie wird durch die Funktion $dist : LS \times LS \rightarrow \mathcal{N}^+$ repräsentiert. Diese wird z.B. als Zeit interpretiert. Der unmittelbare Nachfolger jeder Aufgabe $a \in AS$ innerhalb derselben Tour wird durch die Variable a_{succ} mit $dom(a_{succ}) = AS \backslash \{a\}$ bestimmt. Aufgaben, die innerhalb der gleichen Tour stattfinden, müssen überlappungsfrei angeordnet werden. Für eine Aufgabe a_i und einen potentiellen Nachfolger $a_{i_{succ}}$ von a_i gilt:

$$\begin{aligned} \text{Sei } a_{i_{succ}} = a_j : a_{i_{res}} = a_{j_{res}} \wedge a_{i_{start}} + dur(a_i) + dist(loc(a_i), loc(a_j)) \leq \\ a_{j_{start}} \quad i, j \in \{1, ..., n\}, i \neq j \end{aligned} \tag{2.15}$$

Die erste Aufgabe $a_{k,s}$ und die letzte Aufgabe $a_{k,e}$ einer Tour ist jeweils eine Pseudo-Aufgabe, die das Depot markiert, d.h.:

$$\begin{aligned} \exists a_i.\ a_{k,s_{succ}} = a_i, \quad \forall k \in \{1, ..., u\}, i \in \{1, ..., n\} \\ \exists a_j.\ a_{j_{succ}} = a_{k,e}, \quad \forall k \in \{1, ..., u\}, j \in \{1, ..., n\} \end{aligned} \tag{2.16}$$

Ziel des VRPTW ist die Minimierung der Gesamtfahrzeit über allen Fahrzeugen, die sich in Abhängigkeit von der gewählten Reihenfolge aus der Summe der Fahrzeiten zwischen allen Aufgaben ergibt:

$$\min \sum_{\forall i,j, a_{i_{succ}} = a_j} dist(loc(a_i), loc(a_j)) \tag{2.17}$$

Auch die Minimierung der Gesamtdauer zur Durchführung aller Fahrzeugtouren kann als Optimierungsziel verwendet werden (Gleichung 2.12). Die vorliegende Problembeschreibung basiert auf Kallehauge u. a. (2005). Eine Übersicht über weitere Flottenplanungsprobleme bieten Toth und Vigo (2002).

Weitere Planungsprobleme

Es existieren weitere Problemklassen, die der allgemeinen Modellstruktur gehorchen. Sie unterscheiden sich im Wesentlichen in ihrer Interpretation von Aufgaben

und Ressourcen.
Bei einem *Stundenplanungsproblem* an der Universität werden die Aufgaben als Lehrveranstaltungen und die Ressourcen als Räume interpretiert. Der Planungshorizont wird für jeden Unterrichtstag in eine bestimmte Anzahl von Blöcken mit einer Dauer von je 90 Minuten unterteilt. Jeder Lehrveranstaltung müssen in jeder Woche des Semesters die erforderliche Anzahl von Blöcken und die benötigten Ressourcen zugeteilt werden. Die Aufgaben können zu Aufgabengruppen zusammengefasst, d.h. aggregiert werden. So können z.B. Lehrveranstaltungen (Vorlesungen, Seminare, usw.) nach ihrer Zugehörigkeit zu einem bestimmten Modul gruppiert werden. Auch die verfügbaren Räume können nach ihrer Ausstattung (PC-Kabinett, Labortyp, Anzahl der Plätze) oder nach ihrer Lage (Stockwerk, Gebäude) gruppiert werden, sodass auch aggregierte Planungsentscheidungen möglich sind. Um einen gültigen Stundenplan zu erhalten, müssen eine Reihe von Randbedingungen berücksichtigt werden, wie z.B.:

- „Lehrveranstaltungen, die zum gleichen Modul gehören, dürfen nicht zur gleichen Zeit stattfinden."
- „Lehrveranstaltungen, die dem gleichen Raum zugeordnet sind, dürfen nicht gleichzeitig stattfinden."
- „Lehrveranstaltungen, die vom gleichen Dozenten abgehalten werden, dürfen nicht gleichzeitig stattfinden."

Die Qualität eines Stundenplans wird in der Regel dadurch bestimmt, wie viele Präferenzen bezüglich der Vergabe von Blöcken und Räumen eingehalten werden. Präferenzen können dabei als *weiche Randbedingungen* betrachtet werden, die in einem gültigen Plan nicht eingehalten werden müssen, aber deren Einhaltung die Qualität des Plans verbessert. Es kann z.B. die Präferenz gelten:

- „Bei Veranstaltungen mit mehreren Vorlesungen pro Woche sollten die Vorlesungen nicht am gleichen Tag stattfinden, außer es ist erwünscht."

Eine vollständige, formale Beschreibung eines Stundenplanungsproblems für die Universität ist bei Höckner (2011) zu finden. Brucker und Knust (2001) zeigen, dass Stundenplanungsprobleme als Variante eines RCPSP formuliert werden können.
Bei einem *Schichtplanungsproblem* im Krankenhaus werden die Aufgaben als Schich-

ten und die Ressourcen als Krankenpfleger und Krankenschwestern interpretiert. Für unterschiedliche Tageszeiten gelten unterschiedliche Schichttypen (Nachtschicht, Frühschicht usw.). Sie stellen unterschiedliche Anforderungen an die Anzahl und Qualifikation der eingesetzten Pfleger. Bei der Zuteilung von Pflegern zu Schichten müssen zudem die Arbeitszeitgesetze eingehalten werden. Diese regeln die Abfolge der Schichten, sodass z.B. Nachtschicht und Frühschicht für einen Pfleger nicht direkt aufeinander folgen dürfen und nur eine begrenzte Anzahl von Nachtschichten in einer Woche geleistet werden darf. Eine formale Beschreibung eines Schichtplanungsproblems für ein Krankenhaus ist bei Qu und He (2009) zu finden. Weitere Schichtplanungsprobleme werden von Ernst u. a. (2004) beschrieben.

2.3 Verfahren zur Lösung von strukturierten Planungsproblemen

Planungsprobleme, die durch formale Modelle beschrieben sind, können mit Hilfe von Algorithmen gelöst werden. In diesem Abschnitt werden die Eigenschaften und Auswahlkriterien wesentlicher Algorithmen untersucht, die in der Planung eingesetzt werden. In Abschnitt 2.3.1 werden zunächst allgemeine Eigenschaften von Lösungsalgorithmen identifiziert. Anschließend wird in Abschnitt 2.3.2 untersucht, wie sich die Eigenschaften von Planungsproblemen, insbesondere deren Komplexitätsklasse, auf den Ablauf und die Resultate von Lösungsalgorithmen auswirken können. In den Abschnitten 2.3.3 bis 2.3.6 werden häufig eingesetzte Verfahren vorgestellt, die auf Erzeugungsheuristiken, Verbesserungsheuristiken und vollständiger Suche basieren. Abschnitt 2.3.3 vermittelt dabei zunächst einen Überblick über diese Verfahren und vergleicht, welche Merkmale sie in Abhängigkeit von den Eigenschaften der betrachteten Planungsprobleme aufweisen. Im letzten Abschnitt wird untersucht, welche zusätzlichen Herausforderungen praktische Planungsprobleme häufig mit sich bringen.

2.3.1 Allgemeine Eigenschaften von Lösungsverfahren

Ein *Lösungsverfahren* versucht, eine oder mehrere Lösungen zu finden. Darüber hinaus haben *Optimierungsverfahren* das Ziel, Lösungen zu ermitteln, die optimale

(minimale oder maximale) oder annähernd optimale Zielfunktionswerte besitzen (Burke und Kendall, 2005).
Die in der Praxis eingesetzten Lösungs- und Optimierungsverfahren unterscheiden sich hinsichtlich folgender Eigenschaften:

Laufzeit: Die Algorithmen können nach ihrer asymptotischen Laufzeitkomplexität im schlechtesten Fall (worst-case) eingestuft werden. Typische Komplexitätsklassen sind dabei die lineare Komplexität $\mathcal{O}(n)$, die polynomiale Komplexität $\mathcal{O}(n^k)$ für $k \geq 1$ und die exponentielle Komplexität $\mathcal{O}(d^n)$ für $d \geq 1$, wobei $n \geq 0$ für die Größe der Eingabedaten steht.
Algorithmen mit gleicher Laufzeitkomplexität können sich hinsichtlich der für konkrete Probleminstanzen gemessenen tatsächlichen Laufzeit unterscheiden. Zum Vergleich der Laufzeiten werden oft Benchmarkinstanzen verwendet. Bei der Beurteilung eines Algorithmus spielt sowohl die durchschnittliche Laufzeit als auch die Streuung der Laufzeiten für eine bestimmte Menge von Probleminstanzen eine Rolle (Gomes u. a., 2000).

Lösungsgarantie: Ein Algorithmus, der eine Lösungsgarantie bietet, liefert immer eine Lösung, wenn für das Problem eine Lösung existiert. Andernfalls liefert der Algorithmus die Aussage, dass keine Lösung existiert. Algorithmen, die keine Lösungsgarantie bieten, finden u.U. keine Lösung, auch wenn eine Lösung existiert. Sie können demzufolge auch nicht mit Sicherheit feststellen, dass für ein bestimmtes Problem keine Lösung existiert.

Optimalitätsgarantie: Ein Algorithmus, der eine Optimalitätsgarantie bietet, liefert stets eine optimale Lösung in Bezug auf die Zielfunktionen (bei CSOPs).

Qualität: Algorithmen ohne Optimalitätsgarantie können danach beurteilt werden, welche Qualität die ermittelten Lösungen in Bezug auf die Zielfunktionen besitzen (bei CSOPs). Ein wichtiges Kriterium ist dabei z.B. die Streuung der Qualität für unterschiedliche Probleminstanzen.

2.3.2 Allgemeine Eigenschaften kombinatorischer Planungsprobleme

Bei der Auswahl bzw. Implementierung eines Lösungsverfahrens müssen die Komplexitätsklasse und die Lösungsdichte des Problems berücksichtigt werden. Beide Eigenschaften werden im Folgenden für Planungsprobleme erläutert.

Die Komplexitätsklasse von Planungsproblemen

Jedes Planungsproblem kann (wie jedes CSP/CSOP) einer Komplexitätsklasse zugeordnet werden (Ausiello, 1999). Die Komplexitätsklassen unterteilen die Probleme nach der worst-case-Laufzeitkomplexität ihrer Lösungs- bzw. Optimierungsalgorithmen.

- **Klasse P:** Die Menge der Probleme mit *polynomieller Zeitkomplexität*. Für sie existiert ein deterministischer Lösungs- bzw. Optimierungsalgorithmus mit polynomieller Laufzeitkomplexität (worst case), der eine Lösungs- und Optimalitätsgarantie bietet.
- **Probleme mit nicht-polynomieller Zeitkomplexität:** Zur Lösung dieser Probleme sind nur deterministische Algorithmen mit Lösungs- bzw. Optimalitätsgarantie bekannt, die im schlechtesten Fall eine nicht-polynomielle (z.B. exponentielle) Laufzeitkomplexität besitzen. Effiziente Algorithmen mit polynomieller Laufzeitkomplexität sind nur unter Verzicht auf die Lösungs- und Optimalitätsgarantie verfügbar. Es werden folgende Komplexitätsklassen unterschieden:

  NP: Die Menge aller Probleme, die von einer nichtdeterministischen Turingmaschine in polynomieller Zeit gelöst werden können. Die Lösungen für Probleme in NP können in polynomieller Zeit verifiziert werden.

  NP-schwer: Ein Problem gehört zur Klasse NP-schwer, wenn jedes Problem aus NP darauf reduziert werden kann.

  NP-vollständig: Die Menge aller NP-schweren Probleme, die auch zur Klasse NP gehören.

Viele kombinatorische Planungsprobleme gehören aufgrund ihrer exponentiell an-

Problem	**Komplexität bei Erzeugung einer Lösung**	**Komplexität bei Erzeugung einer optimalen Lösung**
MLLSP (ohne Ressourcenkapazitäten)	P (Van Hoesel u. a., 2002)	P (Van Hoesel u. a., 2002)
MLLSP (mit Ressourcenkapazitäten)	P, aber „schwierig“ (Begnaud, Benjaafar und Miller, 2009)	NP-schwer (Begnaud, Benjaafar und Miller, 2009)
RCPSP	P (Brucker und Knust, 2012a)	NP-schwer (Brucker und Knust, 2012a)
JSSP	P (Brucker und Knust, 2012a)	NP-schwer (Brucker und Knust, 2012a)
VRPTW mit beliebig vielen Fahrzeugen	P (Campbell und Savelsbergh, 2004)	NP-schwer (Cordeau u. a., 2007)
VRPTW mit begrenzter Menge an Fahrzeugen	NP-vollständig (Cordeau u. a., 2007)	NP-schwer (Cordeau u. a., 2007)

Tabelle 2.1: Übersicht über die Komplexität verschiedener Planungsprobleme

wachsenden Suchraumgröße (vgl. Beispiel 1.6) zur Klasse der NP-schweren Probleme (Brucker und Knust, 2012a).

Beispiel 2.3: Es wird angenommen, dass ein Computer pro Sekunde 10^9 Operationen durchführen kann. Bei einer Datenmenge $D = 70$ ergibt sich bei polynomieller Laufzeit $\mathcal{O}(n^5)$ eine Rechenzeit von weniger als 2 Sekunden. Bei einer exponentiellen Laufzeit von $\mathcal{O}(2^n)$ benötigt der Computer jedoch mehr als 35000 Jahre.

Planungsprobleme können danach klassifiziert werden, welcher Aufwand jeweils zur Erzeugung einer Lösung und zur Erzeugung einer optimalen Lösung erforderlich ist. Tabelle 2.1 zeigt diesbezüglich verschiedene Planungsprobleme. Darüber hinaus konnten für einige Probleme, bei denen das Lösungs- oder Optimierungsproblem im Allgemeinen zur Klasse NP-schwer gehört, Spezialfälle identifiziert werden, die polynomiell lösbar bzw. optimierbar sind (Klasse P). Für sie wurden spezielle Algorithmen entwickelt (Barták, Salido und Rossi, 2010). So sind z.B. Job-Shop-Scheduling-Probleme mit 2 Maschinen und Bearbeitungszeiten $dur(a), \forall a \in AS$, bei denen die Anzahl verspäteter Aufgaben oder die Gesamtausführungszeit minimiert werden soll, in polynomieller Zeit optimierbar (Kravchenko, 2000).
Polynomiell und nicht-polynomiell lösbare Planungsprobleme wurden von Lawler u. a. (1993) und Prot, Bellenguez-Morineau und Lahlou (2013) zusammengestellt.

Die Lösungsdichte von Planungsproblemen

Die Lösungsdichte (engl. constraint tightness) t einer Randbedingung oder eines Netzwerks von Randbedingungen über n Variablen $x_1, .., x_n$ wird definiert als der Anteil der gültigen n-Tupel an der Anzahl aller möglichen n-Tupel (Lecoutre, 2010). Sei I die Menge aller gültigen n-Tupel und D_i der jeweilige Wertebereich von $x_i, i \in \{1, .., n\}$, so gilt:

$$t = \frac{Anzahl(I)}{Anzahl(D_1 \times D_2 \times ... \times D_n)} \tag{2.18}$$

Es ist auch möglich, dass eine Randbedingung eine Lösungsdichte von $t = 0$ besitzt, da kein gültiges Werte-Tupel existiert. Die Lösungsdichte entspricht der Wahrscheinlichkeit, aus der Menge der Tupel zufällig ein gültiges Tupel herauszugreifen. Eine geringe Lösungsdichte der Randbedingungen eines Planungsproblems deutet darauf hin, dass mehr Instanziierungen erzeugt werden müssen, bevor eine Lösung gefunden wird. Dies kann zu einem höheren Suchaufwand und damit zu einer höheren Rechenzeit führen.

2.3.3 Lösungsverfahren im Überblick

Für die Verarbeitung von Planungsproblemen gibt es folgende Optionen (vgl. Abbildung 2.4):

- *Heuristische Algorithmen:* Sie können in *Erzeugungsheuristiken* und *Verbesserungsheuristiken* unterteilt werden (Pinedo (2012), Kapitel 14). Heuristische Algorithmen betrachten nur einen Ausschnitt des Suchraums. Daraus resultiert in der Regel eine geringe Laufzeit (linear oder polynomiell). Sie können jedoch nur dann eine Lösungsgarantie bieten, wenn die Erzeugung einer Lösung in polynomieller Zeit möglich ist (vgl. Tabelle 2.1). Heuristische Verfahren zielen in der Regel darauf ab, eine Lösung mit einer möglichst guten Qualität bezüglich der Zielfunktionen zu liefern. Wenn die Erzeugung einer optimalen Lösung für das gegebene Planungsproblem zur Klasse NP-schwer gehört, können sie jedoch keine Garantie für das Finden einer optimalen Lösung bzw. für eine bestimmte Lösungsgüte bieten.

- *Exakte Algorithmen (vollständige Suche):* Diese Verfahren betrachten im schlechtesten Fall den gesamten Suchraum. Sie bieten daher eine Garantie

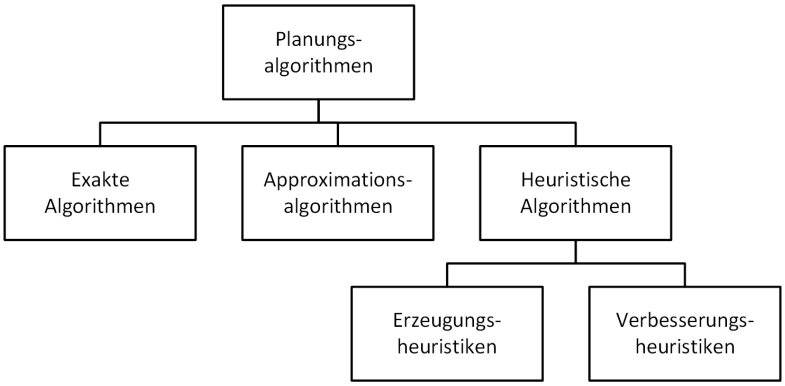


Abbildung 2.4: Klassen von Planungsalgorithmen

dafür, eine Lösung zu finden oder nachzuweisen, dass keine Lösung existiert. Exakte Optimierungsverfahren bieten darüber hinaus eine Garantie dafür, eine optimale Lösung zu finden (wenn eine Lösung existiert). Im schlechtesten Fall weisen exakte Verfahren eine exponentielle Laufzeit auf. Häufig sind sie jedoch so konstruiert, dass sie für Probleminstanzen mit moderater Größe im Durchschnitt wesentlich schneller arbeiten als im schlechtesten Fall. Es werden Techniken eingesetzt, die u.a. auf einer effizienten Einschränkung des Lösungsraums (z.B. mit dem Branch-and-Bound-Verfahren: Görz, Rollinger und Schneeberger (2003), Pinedo (2012), Kapitel 15) oder auf einer Zerlegung in Teilprobleme (z.B. mit dynamischer Programmierung) basieren (Woeginger, 2003). Die tatsächlich benötigte Zeit, um eine Lösung bzw. eine optimale Lösung zu finden, hängt dabei auch von der Lösungsdichte der konkreten Probleminstanz ab.

- *Approximationsalgorithmen:* Diese Verfahren arbeiten in polynomieller Zeit und garantieren darüber hinaus, dass die Qualität der erzielten Lösung maximal um einen bestimmten Prozentsatz von der optimalen Lösung abweicht. Auf Approximationsalgorithmen wird im Folgenden nicht näher eingegangen, da es sich in der Regel um Verfahren handelt, die speziell für einzelne Probleme konstruiert wurden. Es sei auf die Literatur verwiesen (Vazirani, 2001).

In Tabelle 2.2 werden die Eigenschaften heuristischer und exakter Lösungsverfahren zusammengefasst. In den folgenden Abschnitten werden die Grundprinzipien von weit verbreiteten heuristischen und exakten Algorithmen im Detail erläutert.

Verfahren	**Laufzeit**	**Garantie für Lösung?**	**Garantie für optimale Lösung?**
Erzeugungsheuristik	linear, polynomiell	keine Garantie für Probleme in NP	keine Garantie für Probleme in NP, „gute“ Qualität der Lösung wird angestrebt
Verbesserungsheuristik	linear, polynomiell	keine Garantie für Probleme in NP, i.d.R. wird eine Ausgangslösung vorausgesetzt	keine Garantie für Probleme in NP, „gute“ Qualität der Lösung wird angestrebt
exakte Algorithmen (vollständige Suche)	exponentiell im schlechtesten Fall, ggf. deutlich kürzere Laufzeit im durchschnittlichen Fall	ja	ja

Tabelle 2.2: Vergleich der Eigenschaften verschiedener Lösungsverfahren (NP entspricht hier: NP-schwer)

2.3.4 Das Grundprinzip von Erzeugungsheuristiken

Algorithmen mit *Erzeugungsheuristiken* arbeiten nach dem Prinzip, eine partielle Lösung Schritt für Schritt zu erweitern, bis eine vollständige Lösung vorliegt. Die Erweiterung zielt in jedem Schritt darauf ab, die Qualität der Lösung zu maximieren. Daher werden alle Erweiterungsmöglichkeiten zunächst anhand einer Heuristik bewertet, die auf die Zielfunktion des Problems abgestimmt ist. Anschließend wird die beste Möglichkeit ausgewählt (Schneider und Kirkpatrick (2006), Gendreau und Potvin (2005) und Pinedo (2012), Kapitel 14). Algorithmus 1 zeigt ein allgemeines Lösungsschema auf der Basis eines CSOPs[4]. Eine Garantie für die Optimalität der nach diesem Verfahren entstandenen vollständigen Lösung gibt es nicht (vgl. Tabelle 2.2), es wird vielmehr eine Annäherung der optimalen Lösungsqualität angestrebt.
Der Kern des Verfahrens besteht in einer schrittweisen Variablen- und Wertauswahl, die solange ausgeführt wird, bis jeder Variablen ein Wert zugewiesen wurde. Jede Auswahl erfolgt aufgrund einer Heuristik, die sowohl die bisher erstellte Teillösung, als auch die Menge der noch nicht zugewiesenen Variablen berücksichtigen kann. Bei der Wertauswahl für die aktuell betrachtete Variable wird darauf ge-

[4]Verfahren mit Erzeugungsheuristiken können als unvollständiges, informiertes Suchverfahren aufgefasst werden, bei dem im Suchbaum genau ein Pfad von der Wurzel bis einem Blatt konstruiert wird. Die Erweiterung der partiellen Lösung erfolgt nach einer „gierigen“ Strategie, d.h. es wird stets die aus lokaler Sicht beste Auswahl getroffen (Görz, Rollinger und Schneeberger, 2003).

achtet, dass keine Randbedingungen in Verbindung mit den bisher zugewiesenen Variablen verletzt werden. Das Verfahren kann von Vor- und Nachbereitungsschritten umschlossen werden. Vorbereitungsschritte werden z.B. benötigt, um Berechnungen durchzuführen, die eine Voraussetzung für die Anwendung der Heuristiken sind. In Nachbereitungsschritten wird die Lösung ggf. weiter verbessert (Schneider und Kirkpatrick, 2006).

Algorithmus 1 : Lösung eines CSPs mit einer Erzeugungsheuristik

input : Ein CSOP (V, D, C, f), auf das die Heuristik abgestimmt ist.

output : Eine Lösung $assigned = \{\langle v_1, w_1\rangle, ..., \langle v_n, w_n\rangle\}$ mit $n = |V|$, $\forall\langle v, w\rangle \in assigned : v \in V, w \in dom(v)$

$unassigned \leftarrow V$;
$assigned \leftarrow \emptyset$;
$preprocess(unassigned)$;
while $unassigned \neq \emptyset$ **do**
  /*Variablenauswahl aus unassigned:*/
  $v_{select} \leftarrow select\_var\_by\_heuristic(assigned, unassigned)$;
  $D_{select} \leftarrow dom(v_{select})$;
  /*Wertauswahl aus D_{select} unter Beachtung der Randbedingungen:*/
  $value \leftarrow select\_val\_by\_heuristic(D_{select}, assigned, unassigned)$;
  $assigned \leftarrow assigned \cup \{\langle v_{select}, value\rangle\}$;
  $unassigned \leftarrow unassigned \backslash \{v_{select}\}$;
end
$postprocess(assigned)$;
return $assigned$;

Die Variablen- und Wertauswahl erfolgt bei Lösungsverfahren, die nach diesem Vorbild arbeiten, in der Regel in polynomieller Zeit mit einem niedrigen k (häufig gilt $k \leq 3$, vgl. Campbell und Savelsbergh (2004)), z.B. in $\mathcal{O}(n^k)$, wenn n die Anzahl der Variablen ist. Für die Laufzeitkomplexität sind sowohl die in Algorithmus 1 gezeigten Schritte, als auch die von den Funktionen *select_var_by_heuristic* und *select_val_by_heuristic* zur Variablen- bzw. Wertauswahl durchgeführten Schritte (z.B. eine Sortierung von Variablen bzw. Werten nach einer Heuristik) verantwortlich.

Erzeugungsheuristiken werden häufig bei Problemen eingesetzt, bei denen jede Instanziierung eine Lösung ist bzw. wenn die Lösungsdichte t sehr hoch ist. Für

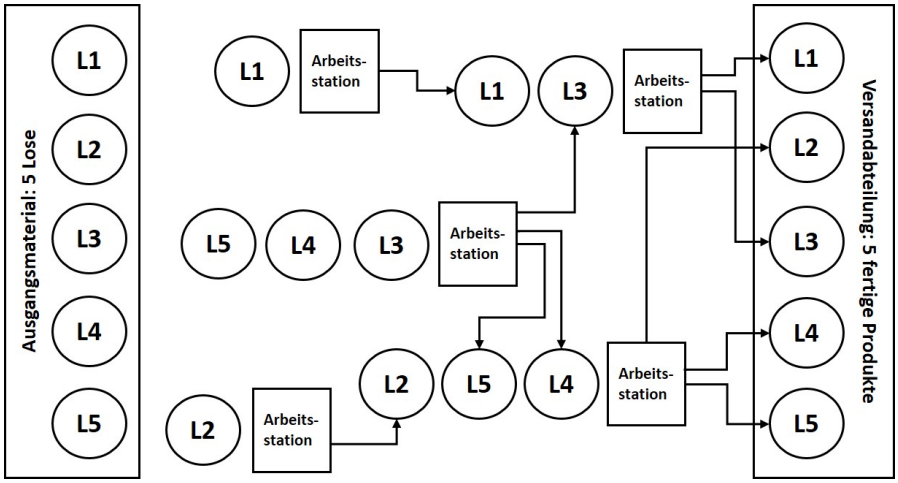


Abbildung 2.5: Verarbeitung von Rohmaterialien zu verschiedenen Produkten. Je nach Bearbeitungssequenz durchlaufen die einzelnen Lose nacheinander bestimmte Arbeitsstationen.

Probleme mit $t < 1$ ist es nicht immer möglich, die partielle Lösung jeweils so zu erweitern, dass keine Randbedingungen verletzt werden (in diesem Fall liefern die Funktionen *select_var_by_heuristic* bzw. *select_val_by_heuristic* den Wert $false$ zurück). Das heuristische Verfahren bricht dann die Lösungssuche erfolglos ab, obwohl u.U. eine Lösung existiert (vgl. Tabelle 2.2). Dieser Fall wird bei vollständigen Suchverfahren berücksichtigt (siehe Abschnitt 2.3.6).

Erzeugungsheuristiken in der Produktionsplanung

In der Produktionsplanung werden Erzeugungsheuristiken häufig als *Verteilungsregel* bezeichnet. Die Ausgangssituation wird in Abbildung 2.5 veranschaulicht, wobei das JSSP zugrunde gelegt werden kann. Ein Produktionsauftrag zur Herstellung eines Produkts oder einer Produktkomponente durchläuft mehrere Bearbeitungsstufen auf bestimmten Maschinen oder Arbeitstationen. Ein Auftrag, der die nächste Bearbeitungsstufe erreicht, reiht sich vor der benötigten Maschine mit anderen Aufträgen in eine Warteschlange ein. Mit Hilfe einer Verteilungsregel werden die Aufgaben in der Warteschlange nach Prioritäten sortiert. Die Aufgabe mit der höchsten Priorität wird als nächstes ausgeführt, sobald die Maschine frei wird (Pinedo (2012), Kapitel 14).
Jede Verteilungsregel ermittelt die Prioritäten aufgrund bestimmter Parameter der Aufgaben oder der Ressourcen oder aufgrund der aktuellen Situation. Einige

bekannte Verteilungsregeln sind z.B. (Pinedo (2012) Kapitel 14, Ingimundardottir und Runarsson (2011), Dominic, Kaliyamoorthy und Kumar (2004) und Panwalkar und Iskander (1977)):

- *Shortest Processing Time (SPT)*: Der Auftrag mit der kürzesten Bearbeitungsdauer wird zuerst ausgeführt.
- *Earliest-Due-Date (EDD):* Der Auftrag a mit dem frühesten Fälligkeitstermin $due(a)$ wird zuerst bearbeitet.
- *Minimum Slack First (MS):* Der Auftrag a mit minimalem Schlupf bis zum spätestmöglichen Ausführungsbeginn $max(due(a) - dur(a) - t, 0)$ wird zuerst bearbeitet. Dabei ist t der aktuelle Zeitpunkt.

Sels, Gheysen und Vanhoucke (2012) liefern eine Übersicht über verschiedene Verteilungsregeln und die mit ihnen hinsichtlich typischer Zielfunktionen erzielten Lösungsqualitäten. Die Auswahl der passenden Regel für eine Ressource kann auch durch sogenannte Hyper-Heuristiken gesteuert werden (Ochoa u. a., 2009). Beispiel 2.4 zeigt, wie dieses Planungsprinzip umgesetzt werden kann. Im Unterschied zu diesem Beispiel erfordern es einige Verteilungsregeln, dass die Sortierung der Aufgaben dynamisch nach jedem Zuweisungsschritt aktualisiert wird (vgl. Ingimundardottir und Runarsson (2011)).

Beispiel 2.4 (Ablauf der Produktionsplanung in ASPROVA):
Das in ASPROVA zugrunde liegende Planungsmodell ist eine Spezialisierung des JSSPs unter Einbeziehung von Rüstzeiten. Für jede Aufgabe stehen mehrere Kombinationen von Ressourcen zur Auswahl. Algorithmus 2 zeigt den Ablauf der Planung (Asprova, 2013): Alle Aufgaben in der Aufgabenmenge AS werden mit dem Prozeduraufruf $dispatch(AS)$ zunächst nach einer vom Nutzer konfigurierbaren Verteilungsregel sortiert. In der dadurch definierten Reihenfolge erfolgt anschließend die Ressourcenzuweisung. Für jede aktuell betrachtete Aufgabe werden alle möglichen Ressourcenkombinationen evaluiert („tentative assignment“). Die beste Kombination wird der Aufgabe zugewiesen, bevor die nächste Aufgabe betrachtet wird.

Algorithmus 2 : Heuristische Planung in ASPROVA

```
Scheduled ← {};
Tasks ← dispatch(AS);
while Tasks ≠ ∅ do
    task ← first(Tasks);
    Scores ← {};
    for candidate ∈ dom(task_res) do
        /*calc_time ermittelt den frühestmöglichen Zeitpunkt, an dem Aufgabe
          task eingeplant werden kann, wenn die in candidate kombinierten
          Ressourcen eingesetzt werden:*/
        tent_start ← calc_time(task, candidate);
        /*Bewerte diesen Zeitpunkt anhand einer Heuristik:*/
        score ← evaluate(task, candidate, tent_start);
        /*Speichere Ressourcen- und Zeitzuweisung unter dem Schlüssel score
          ab:*/
        Scores.add(score, (candidate, tent_start));
    end
    /*Ermittle die beste Ressourcenkombination anhand der Scores:*/
    (best_res, best_start) ← select_best(Scores);
    /*Weise der Aufgabe task die ermittelte Zeit und Ressource zu:*/
    task_res ← best_res;
    task_start ← best_start;
    Tasks ← Tasks\{task};
    Scheduled ← Scheduled ∪ {task};
end
return Scheduled;
```

ASPROVA erlaubt die Definition zusätzlicher Randbedingungen für die Ausführungszeiten der einzelnen Aufgaben, wie z.B. die Einhaltung von Lieferterminen der zugehörigen Aufträge. Bei der heuristischen Planung wird eine Verletzung dieser Randbedingungen in Kauf genommen.

Das Laufzeitverhalten einer Planung mit Verteilungsregeln ist polynomiell. Dabei wird für die Sortierung nach Prioritäten in der Regel ein Aufwand von $\mathcal{O}(n*log(n))$ benötigt (Cormen, 2010). Für das Planungsverfahren in Beispiel 2.4 ist ein Auf-

wand von $\mathcal{O}(n * log(n) + n * d * h)$ erforderlich. Dabei ist d die maximale Domänengröße der Variablen a_{res} (d.h. die maximale Anzahl der Ressourcen, die a zugeordnet werden können). Mit h wird die Größe des Planungshorizontes bezeichnet. Sie entspricht der maximalen Anzahl von Zeitpunkten, die jeweils zur Suche nach einer freien Lücke traversiert werden müssen.

Erzeugungsheuristiken in der Flottenplanung

Bei der Flottenplanung (z.B. nach dem VRPTW-Modell) muss eine gegebene Menge an Aufgaben zu ein oder mehreren Touren angeordnet werden (Abschnitt 2.2.3). Die Erzeugung einer Lösung kann mit Hilfe von Einfüge-Heuristiken erfolgen. Dabei wird in jeder Iteration die aktuelle Teilroute durch Einfügen einer weiteren Aufgabe erweitert. Die beste Einfügeposition wird von einer Heuristik bestimmt. Es ist z.B. typisch, die Position zu wählen, bei der sich die Gesamtfahrzeit der Tour nur minimal verlängert. Wenn aufgrund der Zeitfenster keine weitere Aufgabe mehr in die Tour eingefügt werden kann, wird eine neue Tour angelegt. Das Verfahren erzeugt stets eine Lösung, wenn mit einer unbegrenzten Menge von Fahrzeugen gearbeitet werden kann (Campbell und Savelsbergh, 2004; Solomon, 1987).
Eine weitere Erzeugungsheuristik basiert darauf, die Aufgaben zunächst nach geografischen Gesichtspunkten zu gruppieren (cluster-first-route-second, Laporte u. a. (2000)). Für jede Gruppe wird eine Tour gebildet, die anschließend einem Fahrer zugeteilt wird. Das Verfahren erzeugt nur dann stets eine Lösung, wenn die Zeitfenster der Aufgaben (Gleichung 2.14) vernachlässigt werden können.

2.3.5 Das Grundprinzip von Verbesserungsheuristiken

Algorithmen mit Verbesserungsheuristiken bewegen sich im Lösungsraum L eines Problems. Sie starten mit Ausgangslösungen, die z.B. mit Hilfe von Erzeugungsheuristiken gewonnen werden können, und verbessern diese iterativ. Dabei wird eine Zielfunktion f eingesetzt, mit der die Qualität einer Lösung bewertet werden kann (siehe Abschnitt 2.1). Im Folgenden gilt die Annahme, dass eine Lösung mit einem möglichst kleinen Zielfunktionswert gesucht wird (Minimierungsproblem). Im Allgemeinen basieren Algorithmen mit Verbesserungsheuristiken auf dem Grundprinzip der *lokalen Suche* (Gerdes, Klawonn und Kruse, 2004; Gen-

dreau und Potvin, 2005; Funke, Grünert und Irnich, 2005).
Bei der lokalen Suche werden ausgehend von einer Lösung $L_i \in L$ durch bestimmte Modifikationen Nachbarlösungen erzeugt. Die Menge aller Lösungen, die durch einen Satz von Modifikationen erreicht werden kann, wird als Nachbarschaft $N(L_i) \subset L$ der aktuellen Lösung in Bezug auf diese Modifikationen bezeichnet. Ein Nachbar $L_j \in N(L_i)$ wird als verbessernder Nachbar bezeichnet, wenn gilt: $f(L_j) < f(L_i)$. Wenn eine Lösung L_i keinen verbessernden Nachbarn besitzt, d.h. $f(L_j) \geq f(L_i), \forall L_j \in N(L_i)$, dann ist sie ein lokales Optimum des Lösungsraums. Lokale Suchverfahren starten mit einer Ausgangslösung. In jeder Iteration wird die Nachbarschaft der aktuellen Lösung nach verbessernden Lösungen durchsucht. Verbessernde Lösungen dienen als aktuelle Lösung(en) für die nächste Iteration. Sobald ein lokales Optimum erreicht ist, stoppt die lokale Suche und liefert dieses als Ausgabe (Funke, Grünert und Irnich, 2005).
Zur Erzeugung von Nachbarschaftslösungen werden die Wertzuweisungen der Variablen nach einem bestimmten Schema neu kombiniert. Bei einem Flottenplanungsproblem können zur Optimierung einer Tour z.B. folgende Modifikationen in jeder Iteration eingesetzt werden (Funke, Grünert und Irnich, 2005):

- Tausch der Positionen zwischen jeweils zwei Aufgaben, die in einer Tour direkt nebeneinander liegen.
- Zufällige Neupositionierung einer zufällig ausgewählten Aufgabe in der gleichen Tour.

Ein lokales Suchverfahren soll in der Regel so gestaltet werden, dass das lokale Optimum möglichst nah am globalen Optimum des Lösungsraumes liegt oder mit diesem zusammenfällt (es gibt keine Optimalitätsgarantie, vgl. Tabelle 2.2). Sowohl die Wahl der Ausgangslösung als auch die Wahl der Modifikationen in jeder Iteration hat Einfluss darauf, welche Teile des Lösungsraumes vom Algorithmus untersucht werden, bevor das Verfahren terminiert. Es bietet sich daher an, mehrere diesbezüglich unterschiedlich konfigurierte Instanzen des Algorithmus parallel auszuführen, um eine größere Abdeckung des Lösungsraumes zu erzielen (Prenzel und Ringwelski, 2011).
Das Basisverfahren der lokalen Suche wird auch als einfaches Hillclimbing bezeichnet (Gerdes, Klawonn und Kruse, 2004). Es existiert eine Vielzahl von Abwandlungen des Basisverfahrens, wie z.B. stochastisches Hillclimbing, Simulated Annealing, Tabu-Suche oder Genetische Algorithmen (Gerdes, Klawonn und Kru-

se, 2004). Alle Verfahren gehören zur Klasse der Verbesserungsheuristiken, aber sie unterscheiden sich hinsichtlich ihrer eingesetzten *Metaheuristik* (Gendreau und Potvin, 2005). Eine Metaheuristik ist eine problemunabhängige Strategie, mit der das Vorgehen bei der lokalen Suche gesteuert wird. Sie beschreibt u.a. die Initialisierung des Algorithmus (wie viele und welche Startlösungen?) und die Strategie zum Durchlaufen des Lösungsraums (welche Modifikationen werden eingesetzt und wird eine Verschlechterung der Qualität zugelassen?). Auch die zwischenzeitliche Verarbeitung ungültiger Instanziierungen ist Bestandteil einiger Metaheuristiken. Algorithmen mit Verbesserungsheuristiken werden u.a. bei Ombuki und Ventresca (2004) und Liouane u. a. (2007) (Produktionsplanung), Burke, Li und Qu (2010) (Schichtplanung im Krankenhaus), Burke und Newall (1999) (Stundenplanung) beschrieben.

2.3.6 Das Grundprinzip der vollständigen Suche

Die in Abschnitt 2.3.4 vorgestellten heuristischen Verfahren können bei der Suche scheitern, wenn die Erweiterung einer partiellen Lösung nicht mehr möglich ist. Außerdem bieten sie i.d.R. keine Garantie dafür, eine optimale Lösung zu finden. Der Grund ist, dass keine systematische Betrachtung aller Kombinationen von Wertzuweisungen für die Variablen des CSPs/CSOPs stattfindet. Welche Wertzuweisungen betrachtet werden, hängt stattdessen von den eingesetzten Heuristiken ab (vgl. Algorithmus 1). Im Folgenden soll ein Lösungsschema vorgestellt werden, welches auf einer vollständigen Suche basiert und daher in der Lage ist, die Existenz oder Nicht-Existenz einer Lösung nachzuweisen. Außerdem wird gezeigt, wie durch eine Verarbeitung des Constraint-Netzwerks die vollständige Suche beschleunigt werden kann.

Lösung eines CSPs durch Tiefensuche mit chronologischem Backtracking

Algorithmus 1 zur heuristischen Lösung eines CSPs soll zu einer vollständigen Suche erweitert werden. Algorithmus 3 zeigt die Vorgehensweise basierend auf einer Tiefensuche mit chronologischem Backtracking (Russell und Norvig, 2012). Im Unterschied zu Russell und Norvig (2012) (Kapitel 6, Abbildung 6.5) wird die Konsistenzherstellung dabei zunächst außer Acht gelassen.

Algorithmus 3 : Lösung eines CSPs mit Tiefensuche und Backtracking

input : Ein CSP (V, D, C)

output : Eine Lösung $assigned = \{\langle v_1, w_1\rangle, ..., \langle v_n, w_n\rangle\}$ mit $n = |V|$, $\forall\langle v, w\rangle \in assigned : v \in V, w \in dom(v)$, wenn eine Lösung existiert, **false**, wenn keine Lösung existiert.

```
begin
    unassigned ← V;
    assigned ← ∅;
    preprocess(unassigned);
    return backtrackSolve(assigned, unassigned);
end

backtrackSolve(assigned, unassigned) ≡
if unassigned ≠ ∅ then
    /*Variablenauswahl aus unassigned:*/
     v_select ← select_var_by_heuristic(assigned, unassigned);
    D_select ← dom(v_select);
    while D_select ≠ ∅ do
        /*Wertauswahl aus D_select unter Beachtung der Randbedingungen:*/
         value ← select_val_by_heuristic(D_select, assigned, unassigned);
        /*false bedeutet: es wurde kein Wert gefunden*/
        if value ≠ false then
            D_select ← D_select\{value};
            assigned ← assigned ∪ {⟨v_select, value⟩};
            unassigned ← unassigned\{v_select};
            B ← backtrackSolve(assigned, unassigned);
            if B ≠ false then
                return B;
            else
                assigned ← assigned\{⟨v_select, value⟩};
                unassigned ← unassigned ∪ {v_select};
            end
        else
            D_select ← ∅;
        end
    end
    return false;
else
    return assigned;
end
```

Wie bei Algorithmus 1 werden Schritt für Schritt Variablen ausgewählt und mit Werten belegt. Das Wertauswahlverfahren (*select_val_by_heuristic*) gibt nur Werte zurück, die keine Konflikte mit den bereits zugewiesenen Variablen aufgrund der Randbedingungen entstehen (eine vorausschauende Überprüfung der Domänen der noch nicht zugewiesenen Variablen findet nicht statt). Existiert ein solcher Wert nicht, gibt die Funktion $false$ zurück. Des Weiteren muss sich mit einem ausgewählten Wert der Lösungsprozess fortsetzen lassen, d.h. der rekursive Aufruf von *backtrackSolve*() darf nicht $false$ zurückliefern.
Wenn in der Domäne D_{select} der Variable v_{select} kein Wert mehr enthalten ist, der (in Verbindung mit den bereits vorgenommenen Variablenzuweisungen) alle Anforderungen erfüllt, wird ein *Backtracking*-Prozess[5] durchgeführt: Die Funktion gibt selbst $false$ zurück, sodass die Kontrolle an die aufrufende Funktion zurückgegeben wird. Diese verwaltet die im letzten Schritt betrachtete Variable und weist dieser einen anderen Wert aus ihrer (verbleibenden) Domäne zu oder führt selbst einen Backtracking-Schritt aus. Der Lösungsprozess ist erst abgeschlossen, wenn es gelingt, jeder Variablen einen Wert zuzuweisen oder wenn sich herausstellt, dass keine Lösung existiert.

Beispiel 2.5: Gegeben sind 4 Variablen und ihre Wertebereiche. *a, b, c, d* seien Werte: $x_1 \in \{a, b, c, d\}$, $x_2 \in \{b, c\}$, $x_3 \in \{c, d\}$, $x_4 \in \{a, b, c\}$
Die Randbedingungen lauten: $x_i \neq x_j$, für $i \neq j, i \in \{1, ..., 4\}, j \in \{1, ..., 4\}$
Abbildung 2.6 zeigt, wie Backtracking eingesetzt wird, um eine gültige Instanz zu finden. Die Variablen sollen in der Reihenfolge x_1, x_2, x_3, x_4 zugewiesen werden. Wenn für x_3 in Schritt 3 ein ungünstiger Wert (hier c) gewählt wird, kann x_4 nicht mehr zugewiesen werden. Erst wenn die Entscheidung für x_3 rückgängig gemacht und in Schritt 4 durch einen anderen Wert (jetzt $x_3 = d$) ersetzt wird, kann die Zuweisung erfolgreich fortgesetzt werden.

Algorithmus 3 stoppt, sobald eine Lösung gefunden wurde. Das Verfahren kann so abgewandelt werden, dass systematisch alle Lösungen erzeugt werden oder eine zielgerichtete Suche nach einer optimalen Lösung stattfindet (Hofstedt und Wolf, 2007).

[5] to backtrack (engl.): zurückziehen, zurückverfolgen

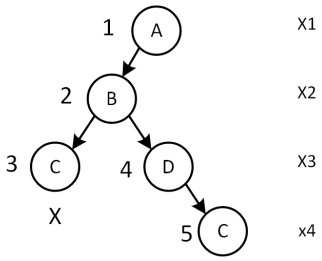


Abbildung 2.6: Backtracking, um eine Lösung für Beispiel 2.5 zu finden

Beschleunigung durch Verarbeitung des Constraint-Netzwerks vor der Suche (k-Konsistenz)

Ein Problem ist k-konsistent, wenn für genau jede Instanziierung von $k-1$ Variablen, die alle Randbedingungen erfüllt, jede beliebige k-te Variable so instanziiert werden kann, dass die k Variablen gemeinsam alle über ihnen existierenden Randbedingungen erfüllen (Freuder, 1982). Bei $k = 1$ wird gefordert, dass kein Wert aus D_i eine unäre Randbedingung über v_i verletzt.

Beispiel 2.6 (Fortsetzung von Beispiel 2.5):
Das Problem ist 2-konsistent: Jede beliebige erste Variable kann mit einem beliebigen Wert aus ihrer Domäne belegt werden, sodass für jede beliebige zweite Variable noch ein gültiger Wert übrig ist. Wenn z.B. $x_2 = c$ gewählt wird, dann verbleibt für x_3 noch der Wert d, für x_4 verbleiben die Werte a oder b und für x_1 die Werte a, b oder d.
Das Problem ist nicht 3-konsistent: Wenn z.B. $x_1 = d$ und $x_2 = c$ gewählt wird, enthält der Wertebereich von x_3 keine gültigen Werte mehr.

Ein Problem ist *stark* k-konsistent, wenn es j-konsistent für alle $1 \leq j \leq k$ ist. Probleme mit n Variablen, die stark n-konsistent sind, werden auch als *global konsistente* Probleme bezeichnet (Freuder, 1982).

Beispiel 2.7: Gegeben sind 4 Variablen und ihre Wertebereiche. Die Buchstaben a bis i seien Werte:
$x_1 \in \{a, b\}$, $x_2 \in \{c, d, e\}$, $x_3 \in \{c, d, e, f\}$, $x_4 \in \{g, h, i\}$
Die Randbedingungen lauten: $x_i \neq x_j$, für $i \neq j, i \in \{1, ..., 4\}, j \in \{1, ..., 4\}$
Das Problem ist 2-, 3- und 4-konsistent. Es ist somit global konsistent.

Bei global konsistenten Problemen ist die Erzeugung einer Lösung mit einem geringen, d.h. linearen Aufwand verbunden, da die Variablen des CSPs in einer beliebigen Reihenfolge mit Werten belegt werden können und stets abgesichert ist, dass die Zuweisung fortgesetzt werden kann. Für nicht global konsistente Probleme ist dagegen meist Backtracking erforderlich (Beispiel 2.6).
Mit Hilfe von Algorithmen kann für jedes CSP vor der Suche ein beliebiges Konsistenzniveau $k \leq n$ hergestellt werden. Bei der Herstellung von k-Konsistenz mit $k > 2$ wird für jedes $(k-1)$-Tupel von Variablen eine Menge von Wertetupeln identifiziert, die eine partielle Lösung darstellen, aber nicht der Bestandteil einer vollständigen Lösung im Lösungsraum sind (Dechter, 2003). Diese inkonsistenten Wertetupel stellen „verbotene Kombinationen“ dar, die durch entsprechende Randbedingungen von der Suche ausgeschlossen werden müssen („nogood-Constraints“, Lecoutre (2013) und Rossi, van Beek und Walsh (2006)). Das durch Hinzufügen solcher Randbedingungen entstandene CSP besitzt eine höhere Lösungsdichte.
Je höher der Wert k für die Herstellung *starker* k-Konsistenz gewählt wird, desto mehr inkonsistente Tupel können identifiziert werden und desto weniger Backtrackingschritte sind bei einem anschließenden Suchvorgang notwendig (Guesgen, 2003). Der Suchvorgang wird dadurch beschleunigt. Bei der Herstellung globaler Konsistenz werden alle inkonsistenten Tupel entfernt. Die Lösungsdichte des daraus resultierenden CSPs beträgt 1.
Allerdings wächst der Aufwand zur Herstellung von (starker) k-Konsistenz polynomiell mit k: $\mathcal{O}(n^k)$. Wenn k zu groß gewählt wird, wird die Beschleunigung der Tiefensuche durch den Aufwand zur Konsistenzherstellung kompensiert. In der Praxis wird daher nur mit geringen Konsistenzniveaus gearbeitet (Guesgen, 2003).

Beschleunigung durch Verarbeitung des Constraint-Netzwerks vor der Suche (lokale Konsistenz)

Häufig fällt es leichter, anstelle von inkonsistenten Tupeln einzelne inkonsistente Werte in den Domänen der Variablen zu identifizieren (wie z.B. bei 1- oder 2-Konsistenz). Aus der Existenz einer bestimmten, n-ären Randbedingung $c_i \in C$ über n Variablen $S_i = \{v_1, ..., v_n\}, v_j \in V, j \in \{1, ..., n\}$ können häufig logische Schlussfolgerungen über ihre Domänen gezogen werden. Angestrebt wird die *lokale Konsistenz*, d.h. ein Zustand, bei dem für jede Variable $v \in S_i$ jeder Wert aus $dom(v)$ gewählt werden kann, sodass für die restlichen Variablen in $S_i \backslash \{v\}$ min-

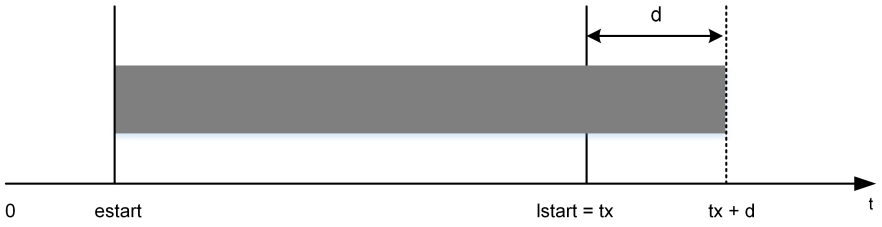


Abbildung 2.7: Besondere Anordnung des Zeitfensters einer Aufgabe a (die Länge des Balkens entspricht der Dauer $dur(a)$): Im Zeitraum von t_x bis $t_x + d$ ist a garantiert aktiv (Beispiel 2.8)

destens eine Wertebelegung existiert, mit der die Randbedingung c_i erfüllt werden kann[6]. Ein Algorithmus, der für ein Constraint c lokale Konsistenz herstellt, wird im Folgenden als *Filteralgorithmus* bezeichnet. Für einige Constraints (z.B. für globale Constraints[7]) werden constraint-spezifische Filteralgorithmen eingesetzt. Sie benutzen die Semantik der jeweiligen Randbedingung, um Schlussfolgerungen zur Reduzierung der Domänen zu ziehen (Beispiel 2.8).

Beispiel 2.8: Die Randbedingungen zur Überlappungsfreiheit aus Gleichung 2.15 des VRPTW-Modells können für jeden Fahrer r zu einem globalen Constraint $nonoverlap(A), A = \{a_i | a_{i_{res}} = r\}$ zusammengefasst werden. Dadurch wird es möglich, einen Filteralgorithmus zu implementieren, der Schlussfolgerungen daraus zieht, wie die Zeitfenster der Aufgaben in einer Tour relativ zueinander angeordnet sind. Beispielsweise kann der Zeitraum $[lstart(a_k), ..., lstart(a_k) + d]$ mit $d = dur(a_k) - (estart(a_k) - lstart(a_k))$ bei allen Aufgaben $a_i \in A, i \neq k$ aus der Domäne von $a_{i_{start}}$ entfernt werden, wenn für eine beliebige Aufgabe $a_k \in A$ gilt: $dur(a_k) > (lstart(a_k) - estart(a_k))$. In diesem Fall ist $[lstart(a_k), ..., lstart(a_k) + d]$ der Zeitraum, indem a_k garantiert aktiv ist, unabhängig davon, ob die Ausführung zum frühest- oder zum spätestmöglichen Termin beginnt (Abbildung 2.7).

Die vollständige Entfernung inkonsistenter Werte ist für viele Randbedingungen

[6]Die lokale Konsistenz wird oft als Verallgemeinerung der 2-Konsistenz mit binären Randbedingungen (Kantenkonsistenz) für n-äre Randbedingungen betrachtet. Lokale Konsistenz wird auch als Hyperkantenkonsistenz oder verallgemeinerte Kantenkonsistenz bezeichnet (Rossi, van Beek und Walsh, 2006).

[7]Wenn mehrere binäre Randbedingungen zu einer gemeinsamen Randbedingung zusammengefasst werden, spricht man von einem *globalen Constraint* (Régin, 2004). Typische globale Constraints sind bei Beldiceanu, Carlsson und Rampon (2005) zu finden.

ein NP-schweres Problem (Bessiere, 2006). In der Praxis werden daher partielle Filteralgorithmen eingesetzt, die nur einen Teil der inkonsistenten Werte identifizieren. Eine umfassende Übersicht über Filtertechniken für Planungsprobleme ist bei (Baptiste, Pape und Nuijten, 2001) zu finden.
Algorithmus 4 zeigt eine Vorgehensweise, um ein CSP, d.h. ein Constraint-Netzwerk, lokal konsistent zu machen. Das Verfahren basiert auf dem AC-3-Algorithmus zur Herstellung von Kantenkonsistenz (Bessiere, 2006). Es gelten die Definitionen aus Abschnitt 2.1.

Algorithmus 4 : Herstellung lokaler Konsistenz in einem CSP mit Hilfe von Filteralgorithmen

input : Ein CSP (V, D, C)

output : Ein CSP' (V', D', C') mit reduzierten Domänen, sodass für alle $D_i \in D, D'_i \in D'$ gilt: $D'_i \subseteq D_i$ *oder* false, wenn das CSP keine Lösung besitzt.

$V' \leftarrow V; D' \leftarrow D; C' \leftarrow C;$
Initialisierung der Warteschlange Q:
$Q \leftarrow C';$
while $Q \neq \emptyset$ **do**
  $c_i \leftarrow select\_and\_remove(Q);$
  $filter \leftarrow get\_filter(c_i);$
  /*entferne inkonsistente Werte aus den Domänen der Variablen in S_i:*/
  $filter(\bigcup_{v \in S_i} dom(v));$
  for *each* $v \in S_i$ **do**
    if $has\_changed(dom(v))$ **then**
      if $dom(v) = \emptyset$ **then**
        return false;
      else
        $Q' \leftarrow$ alle Constraints c_k mit $v \in S_k$;
        $Q \leftarrow Q \cup Q';$
      end
    end
  end
end

Eine bestimmte Randbedingung und der damit verknüpfte Filteralgorithmus wird

genau dann erneut in die Warteschlange Q eingereiht, wenn die Domäne einer von ihr beschränkten Variablen durch die Anwendung eines anderen Filteralgorithmus reduziert wurde. Dies ist notwendig, da durch die Entfernung von Werten die Bedingung der lokalen Konsistenz für diese Randbedingung u.U. nicht mehr erfüllt ist. Da die Reihenfolge, in der die Funktionen ausgeführt werden, nicht fest vorgegeben ist, wird dieser Prozess auch als *chaotische Iteration* bezeichnet.
Wenn die Anwendung der Filter dazu führt, dass eine Variable einen leeren Wertebereich besitzt, steht fest, dass das Problem unlösbar ist. Der Algorithmus wird an dieser Stelle abgebrochen.

Beschleunigung durch Verarbeitung des Constraint-Netzwerks während der Suche (Propagierung)

Algorithmen zur Konsistenzherstellung und Domänenfilterung werden in der Regel nicht nur vor dem ersten Aufruf von *backtrackSolve*, sondern auch nach jedem Zuweisungsschritt angewendet. Eine Zuweisung kann als Randbedingung zur Beschränkung des Wertebereiches einer Variablen auf einen einzigen Wert oder einen bestimmten Teilbereich von Werten aufgefasst werden. Damit verändert jede Zuweisung das Constraint-Netzwerk, sodass die gewählte k-Konsistenz neu hergestellt bzw. die Domänenfilterung aktualisiert werden muss. Die Beschränkung des Wertebereiches einer Variablen hat also im Lösungsprozess zur Folge, dass die Wertebereiche anderer Variablen, die über Randbedingungen mit ihr verbunden sind, ebenfalls reduziert werden müssen (Beispiel 2.9). Dieser Vorgang wird als *Propagierung* bezeichnet, da sich auf diese Weise die Beschränkung im Constraint-Netzwerk ausbreitet (engl. to propagate) (Dechter, 2003; Rossi, van Beek und Walsh, 2006).

Beispiel 2.9 (Propagierung): In Abbildung 2.8 sind die Wertebereiche der Startzeiten s_1, s_2 und s_3 von drei Aufgaben A_1 bis A_3 dargestellt. Es gilt die Randbedingung $s_1 \neq s_2 \neq s_3$. Die Dauer der Aufgaben beträgt jeweils eine Zeiteinheit. Aufgabe A_1 wird die Startzeit 2 zugewiesen. Um 2-Konsistenz zu bewahren, muss die Startzeit 2 aus den Wertebereichen der Aufgaben A_2 und A_3 gestrichen werden. Im zweiten Zuweisungsschritt wird für Aufgabe A_2 die Startzeit 3 festgelegt. Sie wird ebenfalls aus den Wertebereichen der restlichen Aufgaben entfernt, sodass für Aufgabe A_3 nur noch die Startzeit 1 übrig bleibt.

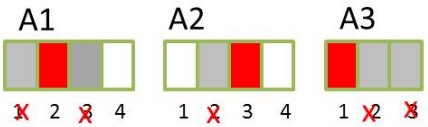


Abbildung 2.8: Propagierung der Startzeit von Aufgaben

Wie stark die Propagierung ist, d.h. wie viele Werte oder Tupel nach jedem Zuweisungsschritt eliminiert werden können, hängt von dem angestrebten Konsistenzniveau bzw. von der eingesetzten Strategie zur Domänenfilterung ab. Es ist das Ziel, in vertretbarer Rechenzeit möglichst viele inkonsistente Werte und Tupel zu identifizieren, um die Suche zu beschleunigen.
Stellt sich heraus, dass nach der Propagierung eine der noch nicht zugewiesenen Variablen einen leeren Wertebereich besitzt, so ist die aktuelle Wertzuweisung (in Verbindung mit den bereits vorgenommenen Variablenzuweisungen) inkonsistent. Sie muss durch einen anderen Wert aus der Domäne ersetzt werden oder es muss ein Backtracking-Schritt eingeleitet werden, wenn keine weiteren Werte mehr zur Verfügung stehen. Ohne Propagierung stellt sich die Inkonsistenz der Wertzuweisung u.U. erst nach zahlreichen weiteren Zuweisungsschritten heraus. Ein Backtrackingvorgang ist dann u.U. mit einem hohen Aufwand verbunden. Die Propagierung ist daher ein wichtiger Bestandteil der Lösungssuche für CSPs. Sie kann durch logisches Schließen Konflikte häufig effizienter aufdecken als die einfache Tiefensuche nach dem Prinzip „Versuch und Irrtum“. Es ist jedoch wichtig, dass Propagierung und Suche in einem sinnvollen Gleichgewicht stehen (vgl. voriger Abschnitt).

Beschleunigung der Suche durch geeignete Variablen- und Wertauswahlstrategien

Auch bei der vollständigen Suche kann die Variablen- und die Wertauswahl durch Heuristiken gesteuert werden (Algorithmus 3). So kann die Reihenfolge, in der die Variablen betrachtet werden, einen großen Einfluss auf den Aufwand der Suche haben. Häufig werden Variablen- bzw. Wertauswahlheuristiken in Verbindung mit einer Propagierung während der Suche eingesetzt. Es haben sich folgende problemunabhängige Heuristiken bewährt (Hofstedt und Wolf, 2007; van Beek, 2006; Guesgen, 2003):

- *Wähle die Variable für den nächsten Zuweisungsschritt aus, deren durch Propagierung reduzierter Wertebereich die kleinste Menge an Werten enthält.*
- *Wähle die Variable für den nächsten Zuweisungsschritt aus, die in den meisten Randbedingungen enthalten ist.*
- *Wähle die Variable für den nächsten Zuweisungsschritt aus, deren Zuweisung mit der größten Wahrscheinlichkeit fehlschlagen wird.*

Jede Variablen- oder Wertauswahlheuristik wird mit dem Ziel eingesetzt, Konflikte in der Variablenbelegung möglichst frühzeitig zu erkennen und damit umfangreichen Backtracking-Schritten vorzubeugen. Dadurch wird die Anzahl an Zuweisungsschritten reduziert und die Suchgeschwindigkeit erhöht (Hofstedt und Wolf, 2007).

Beispiel 2.10 (Fortsetzung von Beispiel 2.6): Es wird angenommen, dass die Werte jeder Variablen in aufsteigender Reihenfolge gemäß lexikographischer Ordnung betrachtet werden. Bei der Zuweisungsreihenfolge x_1, x_2, x_3, x_4 sind 5 Zuweisungsschritte notwendig, bevor eine Lösung gefunden wird. Wenn die Variablen nach der Größe ihrer Wertebereiche sortiert werden, entstehen die Zuweisungsreihenfolgen x_2, x_3, x_4, x_1 oder x_3, x_2, x_4, x_1, bei denen jeweils nur 4 Zuweisungsschritte benötigt werden.

Außerdem helfen die Heuristiken dabei, schnell festzustellen, ob für das Problem keine Lösung existiert (Hofstedt und Wolf, 2007). Die Unerfüllbarkeit kann bereits an einer Teilmenge von Variablen $VS = \{v_1, ..., v_{|VS|}\}, VS \subset V$ festgestellt werden, wenn jede Kombination von Werten der Variablen v_1 bis $v_{|VS|-1}$ bei der Tiefensuche dazu führt, dass anschließend für $v_{|VS|}$ die Funktion $select\_val\_by\_heuristic()$ für alle Werte aus $dom(v_{|VS|})$ stets $false$ zurückliefert. Die Heuristiken zielen darauf ab, eine möglichst kleine Konfliktmenge VS an die Spitze des Suchbaums zu verlegen, damit möglichst wenige Kombinationen von Zuweisungen für den Nachweis der Unerfüllbarkeit betrachtet werden müssen.
Auch die Reihenfolge der Wertauswahl kann die Suchgeschwindigkeit beeinflussen. Eine mögliche Strategie zur Wertauswahl besteht darin, anstelle eines konkreten Wertes in einem Zuweisungsschritt zunächst einen Teilbereich des Wertebereiches auszuwählen (*Domänenreduktion*). Dabei liegt folgendes Prinzip zugrunde (Hofstedt und Wolf, 2007):

- *Verzögere die Werteauswahl solange wie möglich in der Hoffnung, nicht vorzeitig eine falsche Entscheidung zu treffen.*

Eine konkrete Zuweisung kann später vorgenommen werden, nachdem der Teilbereich ggf. durch die Propagierung nachfolgender Zuweisungen weiter reduziert wurde. Auf diese Weise wird verhindert, dass ohne Berücksichtigung der nachfolgenden Zuweisungen vorzeitig ein falscher Wert ausgewählt wird, der später in einem Backtracking-Prozess geändert werden muss.
Wenn die Propagierung nach der Auswahl des Teilbereiches bei anderen Variablen einen leeren Wertebereich verursacht, können alle von ihm umfassten Werte gleichzeitig von der Suche ausgeschlossen werden. Der Aufwand ist dabei wesentlich geringer, als wenn das Scheitern der Zuweisung für jeden einzelnen Wert aus dem Teilbereich festgestellt werden muss.

Anwendungsbeispiele

Die in diesem Abschnitt vorgestellten Lösungsstrategien für CSPs sind die Grundlage sogenannter Constraintlöser, wie z.B. Gecode[8] oder Choco[9]. Anwendungsbeispiele sind das in Fallstudie 2 (Abschnitt 4.3) behandelte Stundenplanungssystem für die Universität, welches mit Choco gelöst wird oder das von Qu und He (2009) mit einem Constraintlöser umgesetzte Schichtplanungsproblem.
Constraintlöser stellen umfangreiche Bibliotheken mit Vorlagen für Randbedingungen und damit verknüpften Filterfunktionen bereit (insbesondere für globale Constraints). Bei der Problembeschreibung kann darauf zugegriffen werden.
Das Grundprinzip der Suche oder die Techniken zur Verarbeitung des Constraint-Netzes können darüber hinaus auch in problemspezifischen und hybriden Lösungsverfahren zum Einsatz kommen, bei denen heuristische und vollständige Suchverfahren miteinander kombiniert werden (Beispiel 2.11).

Beispiel 2.11: Das in Fallstudie 3 (Abschnitt 4.4) behandelte Flottenplanungssystem basiert auf einem Lösungsverfahren, bei dem Erzeugungsheuristiken, Verbesserungsheuristiken und eine vollständige Suche zum Einsatz kommen. Zugrunde liegt ein VRPTW-Modell mit einer begrenzten Fahrzeugmenge (NP-schwer). Das

[8] `www.gecode.org`, abgerufen am 07.01.2016
[9] `http://www.emn.fr/z-info/choco-solver/`, abgerufen am 07.01.2016

Lösungskonzept gestaltet sich wie folgt (Prenzel und Ringwelski, 2011):

1. *Erzeugungsheuristik nach dem Prinzip cluster-first-route-second:* Die Aufgaben werden in geografische Cluster aufgeteilt. Jedem Fahrer wird ein Cluster zugeordnet.

2. *Vollständige Suche mit Einfügeheuristik:* Für jeden Fahrer wird eine Tour als Ausgangslösung erzeugt. Dies erfolgt über eine vollständige Backtracking-Suche, bei der typische Einfügeheuristiken für das VRPTW (vgl. Solomon (1987)) zur Variablen- und Wertauswahl eingesetzt werden.

3. *Verbesserungsheuristik:* Die Qualität der Ausgangslösungen wird anschließend mit einem Hillclimbing-Algorithmus verbessert. Bei entsprechender Konfiguration können für jeden Fahrer auch mehrere Ausgangslösungen nach unterschiedlichen Heuristiken erzeugt und durch Hillclimbing verbessert werden. Die Tour mit der besten Qualität wird dem Fahrer zugeordnet.

4. *Backtracking der Cluster-Zuordnungen:* Wenn die einem Fahrer zugeordneten Aufgaben nicht zu einer Tour angeordnet werden können, müssen einige Aufgaben einem anderen Fahrer zugeordnet werden. Die Auswahl der Aufgaben erfolgt nach einer Heuristik, die die geografische Nähe zu anderen Clustern berücksichtigt.

5. *Aufzeichnung von Konfliktmengen (nogoods):* Wenn der Backtracking-Prozess zur Erzeugung einer Ausgangslösung einer Tour scheitert, wird eine Teilmenge der Aufgaben als Konfliktmenge identifiziert. Es wird verhindert, dass bei einer Änderung der Cluster-Zuordnungen erneut die Aufgaben der Konfliktmenge der gleichen Ressource zugeordnet werden.

Eine Kombination von Constraint-Propagierung und lokaler Suche wird bei De Backer u. a. (2000) beschrieben. Für viele Probleme bietet sich auch eine Kombination mit anderen Optimierungstechniken, wie z.B. der gemischt ganzzahligen Programmierung (**m**ixed **i**nteger **p**rogramming = MIP, Hillier und Lieberman (2001)) an. Eine Kombination dieser Art wird von Timpe (2002) für Grob- und Feinplanungsprobleme angewendet.

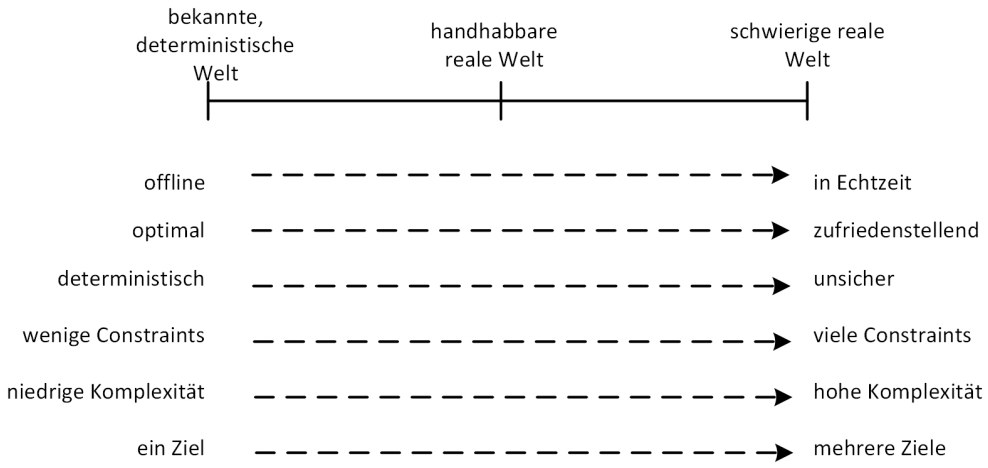


Abbildung 2.9: Das Spektrum von Planungsmodellen (Noronha und Sarma, 1991)

2.3.7 Besondere Anforderungen der Praxis

Aufgrund der in den vorangegangenen Abschnitten vorgenommenen Analyse können die wesentlichen Anforderungen an Planungsalgorithmen wie folgt zusammengefasst werden: Es ist das Ziel, mit vertretbarem Rechenaufwand eine Lösung zu erzeugen, welche alle Randbedingungen des Modells erfüllt und möglichst gute Zielfunktionswerte besitzt. In der Praxis kommen häufig weitere Anforderungen hinzu.

Umgang mit Unsicherheit bezüglich der Planausführung: Es ist häufig unklar, ob die im Modell hinterlegten Bedingungen eingehalten werden können und ob die Aufgaben tatsächlich wie im Plan vorgesehen ausgeführt werden (Cegarra, 2008). Beispielsweise können Ressourcen kurzfristig ausfallen, Aufgaben können sich unerwartet verspäten und neue Aufgaben können nachträglich hinzukommen. Typische Arten von Störungen wurden von Vieira, Herrmann und Lin (2003) zusammengestellt.

Umgang mit Unsicherheit bezüglich des Modells: In einigen Fällen ist es nicht möglich, ein Planungsmodell mit vollständigen Informationen aufzustellen (Cegarra, 2008). So können z.B. die Nachfrage an Produkten, die Dauer der Ausführung von Aufgaben oder die Zeitpunkte, an denen eine Maschine verfügbar ist, oft nicht exakt vorherbestimmt werden.

Berücksichtigung aller Ziele: Die Qualität eines Plans wird häufig nicht nur nach

globalen Leistungsmetriken wie z.B. minimale Kosten bewertet, sondern auch nach lokalen Lösungseigenschaften, d.h. nach der Position einzelner Aufgaben im Plan. Je mehr Präferenzen bezüglich der Lösungseigenschaften erfüllt werden, desto *zufriedenstellender* ist der Plan (Cegarra und Wezel, 2012). Die Präferenzen werden auch als *weiche Randbedingungen* betrachtet[10]. Typische Beispiele sind: „So wenige Vorlesungen wie möglich sollen im 6. Block eines Tages stattfinden“ (Höckner, 2011), „Aufgabe 5 soll vor Aufgabe 6 ausgeführt werden“ (Noronha und Sarma, 1991), „Es sollen so viele Planungswünsche der Krankenschwestern wie möglich erfüllt werden“(Berrada, Ferland und Michelon, 1996). Häufig wird zugunsten der Erfüllung von Präferenzen eine Abweichung vom optimalen Zielfunktionswert bezüglich globaler Metriken in Kauf genommen (Noronha und Sarma, 1991).

Zu Forschungszwecken werden häufig vereinfachte, deterministische Modelle eingesetzt, die Unsicherheiten vernachlässigen und nur wenige Zielfunktionen besitzen. Die in der Praxis benötigten Modelle sind häufig komplexer, enthalten Unsicherheiten und besitzen weitaus mehr Randbedingungen und Zielfunktionen. Zudem wird in der Praxis häufig eine zufriedenstellende Lösung gegenüber einer optimalen Lösung bevorzugt. Abbildung 2.9 zeigt das Spektrum von theoretischen bis zu realistischen Planungsmodellen (Noronha und Sarma, 1991).
Der Umgang mit Unsicherheit erfordert eine Anpassung der Lösungsmethoden für die Praxis. Unsicherheit bezüglich des Modells wird z.B. von stochastischen Planungsmodellen berücksichtigt. Dabei werden unsichere Parameter als Zufallsvariablen repräsentiert (Lei, Laporte und Guo (2011) und Staedler u. a. (2012), Pinedo (2012), Kapitel 9). Unsicherheit bezüglich der Planausführung kann im Voraus durch die Erstellung sogenannter robuster Pläne berücksichtigt werden. Die Anforderungen an die Robustheit können durch geeignete Zielfunktionen oder weiche Randbedingungen ausgedrückt werden (vgl. Xiong, Xing und Chen (2013)). Darüber hinaus ist eine nachträgliche Anpassung des Plans an geänderte Bedingungen häufig unumgänglich (Beispiel 2.12).

Beispiel 2.12 (Ereignisse, die eine Aktualisierung des Plans erfordern):

- Eine reparaturbedürftige Maschine kann nur noch halb so häufig eingesetzt

[10] Im Gegensatz dazu werden Randbedingungen, die in einer gültigen Instanziierung erfüllt sein müssen, als *harte Randbedingungen* bezeichnet.

wie die anderen Maschinen.

- Ein Fahrer hat einem bestimmten Tag 2 Stunden früher Feierabend.
- Ein Auftrag erhält kurzfristig eine höhere Priorität.

Für die Anpassung gelten besondere Anforderungen. Zum einen soll sie häufig in Echtzeit[11] erfolgen, da die Planausführung bereits begonnen hat und eine schnelle Reaktion auf die eingetretenen Ereignisse gefordert wird. Zum anderen sollten die Abweichungen vom ursprünglichen Plan möglichst geringfügig sein, da die Arbeitsumgebung bereits auf den Plan abgestimmt wurde (Kunden wurden über einen Besuch oder eine Lieferung benachrichtigt, Maschinen wurden eingerichtet, Mitarbeiter haben sich auf ihre Schichtpläne eingestellt). Verfahren zur Anpassung von Plänen werden u.a. unter den Stichworten „dynamische Planung“ (z.B. Zandieh und Adibi (2010) und Pillac u. a. (2013)), „reaktive Planung“ (wie z.B. Nie u. a. (2013)) oder „Echtzeit-Planung“ entwickelt (z.B. Hong (2012)). Eine ausführliche Diskussion von Umplanungsverfahren für die Produktionsplanung wird bei Vieira, Herrmann und Lin (2003) vorgenommen.
Planungsprobleme der realen Welt, die zu komplex sind, um innerhalb des geforderten Zeitraums gelöst werden zu können, müssen ggf. durch Vereinfachung in handhabbare Probleme umgewandelt werden (Abbildung 2.9).

2.4 Lösung semi-strukturierter Planungsprobleme mit Hilfe von Expertenwissen

Es wurden Planungsmodelle und Lösungsverfahren vorgestellt, mit denen eine Software zur automatischen Lösung von Planungsproblemen implementiert werden kann. Ein Planungsproblem wird im Folgenden als *strukturiert* bezeichnet, wenn aufgrund der Daten, die zu einem bestimmten Zeitpunkt vorliegen, ohne Beteiligung des Disponenten automatisch ein Plan so erzeugt bzw. aktualisiert werden kann, dass alle Anforderungen erfüllt werden (Abschnitt 1.3.2). In der Praxis gibt

[11] Echtzeit heißt hier: garantiert innerhalb eines bestimmten Zeitraums, der dem Disponenten zur Anpassung des Plans eingeräumt wird. Dieser kann je nach Anwendungsbereich wenige Minuten bis hin zu mehreren Stunden betragen.

es jedoch auch Anforderungen, die nicht im Modell kodiert sind (Cegarra und Wezel, 2012). Solche Anforderungen ergeben sich häufig aus dem Expertenwissen des Disponenten über die Organisation oder die Planungssituation (vgl. Abschnitt 2.4.1). Sie werden im Folgenden als *unstrukturierte Anforderungen* bezeichnet. Planungsprobleme, für die unstrukturierte Anforderungen vorliegen, werden im Folgenden *semi-strukturierte Planungsprobleme* genannt.
In den folgenden Abschnitten soll die Rolle des Disponenten bei der Lösung semi-strukturierter Planungsprobleme untersucht werden. Zunächst werden in Abschnitt 2.4.1 Beispiele für Expertenwissen angegeben, aus dem unstrukturierte Anforderungen resultieren. In Abschnitt 2.4.2 wird ein Modell der Arbeitsumgebung entwickelt, die den Rahmen für die Entscheidungen des Disponenten zur Umsetzung unstrukturierter Anforderungen bildet. Abschnitt 2.4.3 stellt den Prozess der Entscheidungsfindung innerhalb der Arbeitsumgebung als kognitiven Entscheidungsprozess dar.

2.4.1 Gründe und Beispiele für die Anwendung von menschlichem Expertenwissen

Unstrukturierte Anforderungen können aus folgenden Gründen vorliegen:

1. Die Planungssoftware ist nicht individuell auf das Unternehmen oder die Organisation zugeschnitten, sondern wurde für einen breiten Anwenderkreis entwickelt (Standardanwendungen). Einige unternehmensspezifische Parameter, Variablen und Randbedingungen werden daher nicht bei der Planerstellung berücksichtigt.

2. Auch unternehmensspezifische Anwendungen bilden die Anforderungen häufig nicht vollständig ab, da es nicht immer möglich ist, im Prozess der Software-Herstellung alle Anforderungen zu ermitteln (Cegarra, 2008; Crawford u. a., 1999). Ein möglicher Grund ist, dass aus Zeit-, Sicherheits- oder Kostengründen den Vertretern von Software-Firmen nicht genügend Einblick in die Arbeit des Disponenten gewährt wird (McKay und Buzacott, 2000). Zudem fällt es Disponenten häufig schwer, ihr Fachwissen in einer vollständigen und widerspruchsfreien Form für die Software-Entwickler aufzubereiten (Cegarra und Wezel, 2012).

3. Das Planungsmodell wird bewusst gegenüber der Realität vereinfacht, um die Komplexität des Problems zu reduzieren und somit eine Lösungserstellung in akzeptabler Zeit zu ermöglichen (vgl. Abbildung 2.9). Es wird davon ausgegangen, dass der Disponent die zusätzlichen Anforderungen in einer automatisch generierten Ausgangslösung leicht manuell umsetzen kann.

4. Bestimmte Anforderungen können nur schwer in eine strukturierte Form überführt werden, d.h. sie sind nicht kodierbar. Ihre Umsetzung erfordert eine besondere Kenntnis der (aktuellen) Verhältnisse im Unternehmen. Nur der Disponent kann einschätzen, ob ein gegebener Plan zu einem bestimmten Zeitpunkt diese Anforderungen erfüllt oder nicht (Snoo, Wezel und Jorna, 2011; Cegarra und Wezel, 2012).

5. Bestimmte Anforderungen können vom Disponenten nicht klar beschrieben werden. Er bevorzugt aus Erfahrung bestimmte Muster von Planungsentscheidungen, ohne sie konkret begründen zu können. Sie tragen jedoch zu einer erfolgreichen Planausführung und zur Zufriedenheit der Mitarbeiter bei. Es handelt sich um *implizites Wissen* (Burstein und Holsapple, 2008; Cegarra und Wezel, 2011c).

6. Unabhängig von den restlichen Punkten wird das menschliche Urteilsvermögen des Disponenten benötigt, um Unsicherheiten bei Ressourcen, Diensten und Aufgaben zu identifizieren und entsprechende Maßnahmen einleiten zu können (vgl. Abschnitt 1.1.5, Cegarra und Wezel (2012)).

Der Disponent muss bei der Planungerstellung und -aktualisierung sein Expertenwissen anwenden, um die Lücken in der Modellspezifikation zu kompensieren. Im Folgenden werden einige Beispiele für Anforderungen gegeben, die aufgrund der Punkte 1 bis 6 vom Disponenten manuell umgesetzt werden müssen:
Aufgrund von Punkt 1 muss der Disponent häufig die Berücksichtigung unternehmensspezifischer Planungsregeln sicherstellen.

Beispiel 2.13 (unternehmensspezifische Planungsregeln, Snoo, Wezel und Jorna (2011)):

- Fest angestellte Arbeiter sollten möglichst nicht in der gleichen Schicht wie befristet angestellte Mitarbeiter arbeiten.

- Fahrzeiten zwischen zwei Orten dürfen nicht mehr als 30 min. betragen.

Unspezifizierte Anforderungen nach Punkt 2 umfassen häufig inoffizielle Planungsregeln. Der Disponent kennt aus Erfahrung die besten Ausführungszeitpunkte und Ressourcen für bestimmte Aufgaben (Framinan und Ruiz, 2012).

Beispiel 2.14 (inoffizielle Planungsregeln):

- Eine Aufgabe, deren Dauer mehr als 2 Stunden beträgt, sollte nicht zu Beginn des Arbeitstages eingeplant werden.
- Alle Aufgaben für Kunde X sollten von Arbeiter Y ausgeführt werden.

Die Planung beruht häufig auf unsicheren Informationen über Aufträge und Ressourcen (Abschnitt 2.3.7). Wenn aufgrund von Punkt 3 die Anforderungen an die Robustheit nicht vollständig formal spezifiziert werden können, muss der Disponent eine vorausschauende Planung manuell sicherstellen (Cegarra, 2008; McKay und Wiers, 1999; Morikawa und Takahashi, 2006; McKay und Buzacott, 2000). Es sind Pläne gefordert, die flexibel für nachträgliche Änderungen sind, sodass kurzfristig Aufträge oder Ressourcen hinzugefügt, entfernt oder geändert werden können.

Beispiel 2.15 (vorausschauende Planung): In einer Produktionsfirma legt ein Disponent die Maschinenbelegung für den nächsten Monat fest. In den letzten beiden Wochen des Monats nutzt er bewusst nicht die volle Kapazität der Maschinen aus. Auf diese Weise kann er am Anfang des Monats noch kurzfristig neue Aufträge entgegennehmen und einplanen - ein Service, den einige Kunden sehr schätzen.

Disponenten überprüfen erstellte und modifizierte Pläne ständig hinsichtlich ihrer Machbarkeit (Punkt 6). Laut McKay und Wiers (1999) kommen teilweise mehr als 100 Heuristiken zum Einsatz, um potenzielle Konflikte aufzudecken und vorbeugende Maßnahmen zu treffen.

Beispiel 2.16 (vorausschauende Planung): Der Disponent einer Produktionsfirma erwartet eine Materiallieferung, die für nächste Woche angekündigt ist. Bei einer pünktlichen Lieferung könnte die Verarbeitung in der darauffolgenden Woche erfolgen. Da sich der Lieferant erfahrungsgemäß verspätet, plant sie der Disponent jedoch erst in der übernächsten Woche ein.

Auch die Rücksichtnahme auf die menschlichen Ressourcen spielt bei der Planung eine wichtige Rolle. Um die Akzeptanz eines Plans durch die Mitarbeiter sicherzustellen, berücksichtigt der Disponent Anforderungen, die zum Teil nur schwer formalisierbar sind (Punkte 4 und 5). Ihre Umsetzung erfordert Planungsentscheidungen, die für den jeweiligen Planungszeitraum und für die aktuelle betriebliche Situation spezifisch sind.

Beispiel 2.17 (Berücksichtigung von Mitarbeiterwünschen):

- *Planänderungen sollten nachvollziehbar sein.*
  Nachträgliche Planänderungen sollten den Mitarbeitern eine einfache Umstellung ermöglichen. Der Disponent muss abwägen, welche Mitarbeiter in welchem Umfang von einer Änderung betroffen sein sollen und diese rechtzeitig bekanntgeben.

- *Pläne sollten akzeptabel sein, Wünsche der Belegschaft sollten berücksichtigt werden.*
  Der Disponent beeinflusst mit seinen Entscheidungen die Arbeitsbedingungen der Mitarbeiter. Er ist dafür mitverantwortlich, die Zufriedenheit und die Arbeitsmoral der Mitarbeiter zu erhalten. Die individuellen Mitarbeiterwünsche müssen daher bei der Planung mitberücksichtigt werden.

- In einzelnen Anwendungsbereichen existieren noch weitere Anforderungen, wie z.B.: *Die Aufgaben sollten möglichst abwechslungsreich verteilt werden.* (Snoo, Wezel und Jorna, 2011).

Weitere typische unternehmensspezifische Anforderungen an die Qualität von Plänen sind bei Snoo, Wezel und Jorna (2011) zu finden.

Abbildung 2.10: Die Abstraktionshierarchie zur Modellierung der Arbeitsumgebung

2.4.2 Planungsentscheidungen im Kontext der Arbeitsumgebung

Im Folgenden wird das Konzept der Abstraktionshierarchie vorgestellt und auf Planungsprobleme angewendet.

Die Abstraktionshierarchie einer Arbeitsumgebung

Die Abstraktionshierarchie hat sich als arbeitswissenschaftliches Modellierungskonzept bewährt (Vicente, 1999; McIlroy und Stanton, 2011). Bei diesem Konzept wird darauf verzichtet, einzelne Arbeitsaufgaben zu identifizieren. Stattdessen wird die Arbeitsumgebung betrachtet, die einen begrenzten Handlungsspielraum zum Erreichen bestimmter Ziele bereitstellt.
Abbildung 2.10 zeigt das Modell der Arbeitsumgebung als Abstraktionshierarchie. Die Grundlage der Arbeitsumgebung wird von den physikalischen Ressourcen (Maschinen, Arbeiter) gebildet. Sie bieten die funktionale Voraussetzung, um die Unternehmensziele zu erreichen. Sie stellen *Dienste* zur Verfügung, wie z.B. bestimmte Fähigkeiten oder Kapazitäten. Das Unternehmen greift auf sie zurück, um bestimmte *Funktionen*, wie z.B. den Ablauf eines Herstellungsprozesses, zu realisieren. Damit werden letztendlich die *Ziele* der Arbeitsumgebung, wie z.B. die Herstellung von Produkten, erreicht (vgl. Vicente und Rasmussen (1992)).
Jede Ebene der Abstraktionshierarchie zeigt die Arbeitsumgebung aus einem bestimmten Blickwinkel. Von den Eigenschaften der darunter liegenden Ebenen wird jeweils abstrahiert, bzw. werden sie als Voraussetzung angenommen. So werden z.B. auf einer Funktionsebene nur die Details der Funktionsausführung betrach-

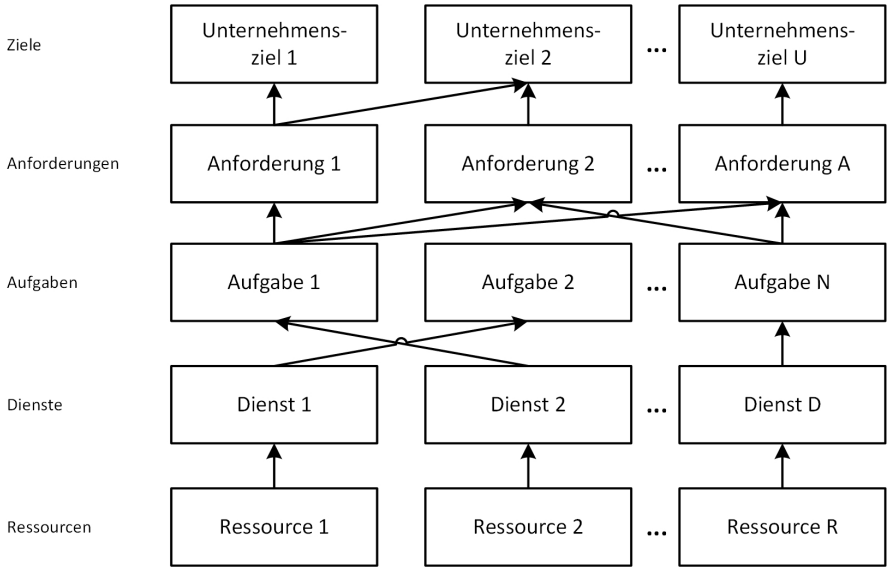


Abbildung 2.11: Die Abstraktionshierarchie für Dispositionsprobleme.

tet. Die Eigenschaften der dafür benötigen Dienste werden vernachlässigt. Es können beliebig viele Ebenen eingesetzt werden, um Dienste, Funktionen und Ziele auf unterschiedlichen Abstraktionsebenen zu modellieren. So können z.B. mehrere, einander übergeordnete Funktionsebenen erstellt werden.
Auf jeder virtuellen Ebene der Arbeitsumgebung steht ein Handlungsspielraum zur Verfügung, der genutzt werden kann, um die Voraussetzungen für die nächste Ebene zu erfüllen. Dieser ist durch Randbedingungen genau so begrenzt, dass nicht mehr und nicht weniger als die auf der obersten Ebene gesetzten Ziele erreicht werden können. Ein Unternehmen sucht z.B. bei der taktischen Planung die minimale Anzahl erforderlicher Maschinen, um ein Produktionsziel zu erreichen.

Eine allgemeine Abstraktionshierarchie für Planungsprobleme

In der Literatur sind Abstraktionshierarchien für ein Routenplanungsproblem (Gacias, Cegarra und Lopez, 2012) und für die allgemeine Produktionsplanung (Higgins, 2001; Higgins, 1998) zu finden. In Anlehnung an die von Higgins (2001) vorgestellte Variante soll in dieser Arbeit eine allgemeine Abstraktionshierarchie für alle Planungsprobleme eingeführt werden. Wie in Abbildung 2.11 dargestellt[12], besitzt sie 5 Ebenen:

[12]Die Verbindungen zwischen den einzelnen Ebenen wurden willkürlich gewählt.

Ressourcen: Es handelt sich um alle Personen, Fahrzeuge und Maschinen, die an der Ausführung eines Plans beteiligt sind. Die Ermittlung der benötigten Ressourcen ist Gegenstand der strategischen und taktischen Planung.

Dienste: Ressourcen stellen Arbeitszeiten, Kapazitäten oder Qualifikationen zur Verfügung. Da die Dienste Kosten verursachen, werden sie von der Organisation auf das Maß begrenzt, das zur Erfüllung der Aufgaben notwendig ist (taktische Planung).

Aufgaben: Sie sind der funktionale Zweck, für den die Dienste eingesetzt werden. Bei der Disposition werden die Dienste zur Ausführung der Aufgaben eingeteilt. Dabei ist es das Ziel, alle Anforderungen zu erfüllen.

Anforderungen: Die in Abschnitt 2.4.1 zusammengefassten Anforderungen stellen sicher, dass die Aufgaben so ausgeführt werden, dass die langfristigen Unternehmensziele erreicht werden.

Ziele: Langfristige Unternehmensziele, wie z.B. Gewinnmaximierung, Kostensenkung, Erhöhung der Kundenzufriedenheit.

Der Dispositionsprozess (operative Planung) lässt sich anhand dieser Darstellung wie folgt beschreiben: Auf der Aufgabenebene werden vom Disponenten Planungsentscheidungen definiert, um die Ziele auf der Anforderungsebene zu erreichen. Dazu gehört die Festlegung der Reihenfolge, der Zeitpunkte und der Dienste zur Ausführung der Aufgaben. Dabei werden ggf. mehrere Aggregationsstufen durchlaufen (Abschnitt 2.2). Die Aufgabendurchführung muss außerdem auf der Dienstebene sichergestellt werden. Dies erfolgt ebenfalls über Entscheidungen, mit denen die ausgewählten Dienste zu den festgelegten Zeitpunkten exklusiv für die Ausführung der Aufgaben gesperrt werden (Abbildung 2.12). Planung findet somit auf der Dienstebene und der Aufgabenebene statt.

Beispiel 2.18 (Einstellungen eines Dienstes): Eine Druckmaschine besitzt 6 Farbwerke. Die benötigten Farben können für jeden Druckauftrag frei gewählt werden. Vor jedem Auftrag müssen die Druckzylinderhülsen in die benötigten Farbwerke eingesetzt werden.

Ursprünglich wurde die Abstraktionshierarchie für Anwendungsbereiche mit sta-

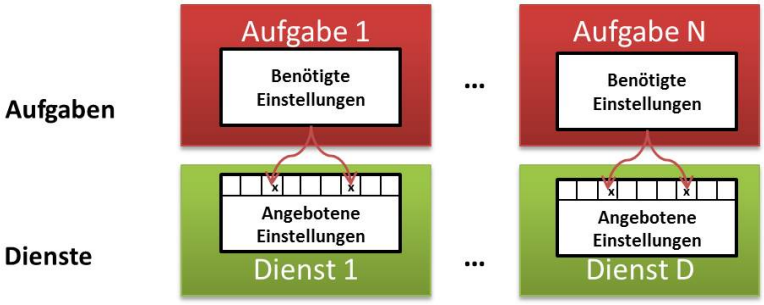


Abbildung 2.12: Festlegung von Entscheidungen auf der Dienstebene, um die Aufgabenausführung sicherzustellen

tischen Funktionen und Zielen entwickelt[13]. Bei der Planung hängt die Funktion der Arbeitsumgebung jedoch von den Aufgaben ab, die gerade ausgeführt werden. Sie ändert sich somit dynamisch in Abhängigkeit von der aktuellen Nachfrage an Produkten und Dienstleistungen. Higgins (2001) betrachtet die Abstraktionshierarchie für Planungsprobleme daher als Sonderfall.
In der Praxis treten häufig Störungen auf, die eine nachträgliche Modifikation von Plänen erfordern. Dabei wird auf der Aufgaben- und Dienstebene eine Umplanung durchgeführt, um trotz der Störung noch alle Anforderungen zu erfüllen.
Eine Aufweitung (engl. constraint relaxation) der harten Randbedingungen von Diensten und Ressourcen gehört in einigen Fällen ebenfalls zum Handlungsspielraum des Disponenten (Cegarra, 2008; Higgins, 1999). Dabei werden die Dienste oder Ressourcen neu konfiguriert, um trotz der Störung die notwendigen Voraussetzungen zur Durchführung der Aufgaben zu schaffen. Auf der Dienstebene ist z.B. eine geringfügige Überschreitung von Kapazitäten (z.B. durch die Anordnung von Überstunden) oder die Änderung von Urlaubsplänen denkbar. Auf der Ressourcenebene kann der Disponent z.B. die Anstellung zusätzlicher Mitarbeiter veranlassen. Diese Maßnahmen werden in dieser Arbeit nicht näher behandelt.

2.4.3 Entscheidungsprozesse im Rahmen der Abstraktionshierarchie

Bei semi-strukturierten Planungsproblemen ist der Disponent an der Spezifikation von Lösungseigenschaften mit beteiligt (Abschnitt 2.4.1). In diesem Abschnitt

[13] wie z.B. (Dadashi u. a., 2013)

wird untersucht, wie der Disponent die Lösungseigenschaften im Kontext der Arbeitsumgebung ermittelt. Dazu wird zunächst das Konzept der wissensbasierten Planung eingeführt und von der vollautomatischen, regelbasierten Planung abgegrenzt. Anschließend wird die wissensbasierte Planung als mehrstufiger Entscheidungsprozess definiert, bei dem die Stufe der Entwurfsphase eine zentrale Stellung einnimmt.

Regelbasierte und Wissensbasierte Planung

Planungsentscheidungen können mit Hilfe von heuristischen und exakten Algorithmen gefunden werden. Dieses Vorgehen wird im Folgenden als *regelbasierte Planung* bezeichnet: Die Lösung wird immer durch Abarbeiten vorbestimmter Regeln oder algorithmischer Anweisungen erzeugt. Zunächst wird das Planungsproblem anhand der Aufgaben und Ressourcen eingeschätzt, anschließend erfolgt die Auswahl und Konfiguration einer Planungstechnik, die geeignet erscheint, wie z.B. die Wahl eines Optimierungsverfahrens und die Gewichtung von Zielfunktionen. Schließlich wird die ausgewählte Technik angewendet, um das Planungsproblem zu lösen.
Eine vollständig regelbasierte Planung ist nur möglich, wenn das Planungsproblem strukturiert ist (Cegarra und Wezel, 2012). In diesem Fall lassen sich mit den verfügbaren Regeln Pläne erzeugen, die die *strukturierten* Anforderungen erfüllen. Sie führt jedoch nicht zum Ziel, wenn auch unstrukturierte Anforderungen vorliegen. Die Umsetzung unstrukturierter Anforderungen erfolgt über geeignete Planungsentscheidungen, die für einzelne Aufgaben im Kontext der Abstraktionshierarchie getroffen werden (Abschnitt 2.4.2). Diese können auf zwei Wegen in den Planungsprozess eingebracht werden:

- *Überführung unstrukturierter in strukturierte Anforderungen:* Dazu muss das Planungsmodell modifiziert werden. Das erweiterte Planungsproblem kann anschließend regelbasiert gelöst werden.

- *Manuelle Modifikation einer automatisch erzeugten Lösung:* Eine nachträgliche Änderung der Wertzuweisungen von einzelnen Variablen in einer Lösung.

Wenn die durch taktische Entscheidungen vorgegebenen harten Randbedingungen nicht verletzt werden dürfen, muss der Lösungsraum L' eines modifizierten Pro-

blems $CSOP'$ eine Teilmenge des Lösungsraums L des ursprünglichen Problems $CSOP$ sein (Abschnitt 1.3.4):

$$L' \subseteq L \tag{2.19}$$

Weiterhin wird davon ausgegangen, dass der Disponent bei der Planung einzelne Aufgaben oder Aufgabengruppen betrachtet (vgl. Abschnitt 2.4.2). Planungsentscheidungen beziehen sich daher auf einzelne Variablen, die bestimmten Aufgaben bzw. Aufgabengruppen zugeordnet sind (z.B. a_{start}, a_{res}, vgl. Abschnitt 2.2) und schränken deren Wertebereich auf eine Teilmenge des ursprünglichen Wertebereiches (häufig auf einen einzigen Wert) ein. Bei der Überführung unstrukturierter Anforderungen in strukturierte Anforderungen gilt für $CSOP'$, welches aus $CSOP = (V, D, C, f)$ hervorgeht, unter Beachtung von Gleichung 2.19:

$$CSOP' = (V, D', C, f) \wedge \forall D'_i \in D', D_i \in D : D'_i \subseteq D_i \tag{2.20}$$

Der Prozess zur Umsetzung unstrukturierter Anforderungen soll im Folgenden als *wissensbasierte Planung* bezeichnet werden. Regelbasiertes und wissensbasiertes Verhalten sind neben dem Reflexverhalten, bei dem die Suche nach der passenden Regel entfällt, Bestandteile des 3-Ebenen-Handlungsmodells von Rasmussen (1983).

Wissensbasierte Planung als Entscheidungsprozess

In vielen Fällen kann Expertenwissen nicht direkt in passende Planungsentscheidungen übersetzt werden (Berry, Peintner und Yorke-Smith, 2007). Häufig ist nicht von vornherein bekannt, mit welchen Entscheidungen eine Anforderung am besten umgesetzt werden kann. In diesem Fall muss der Disponent alternative Entscheidungen aufstellen und mit Hilfe der Abstraktionshierarchie bewerten. Folgende Kriterien fließen in die Bewertung ein:

- Wie gut wird die unstrukturierte Anforderung durch diese Entscheidungen repräsentiert?
- Können alle Randbedingungen der Ressourcen und Dienste erfüllt werden, wenn das Planungsmodell bzw. die Lösung um diese Entscheidungen erweitert wird?
- Wie gut ist der Kompromiss, der mit dem modifizierten Modell bzw. der modifizierten Lösung zwischen den bereits existierenden strukturierten und

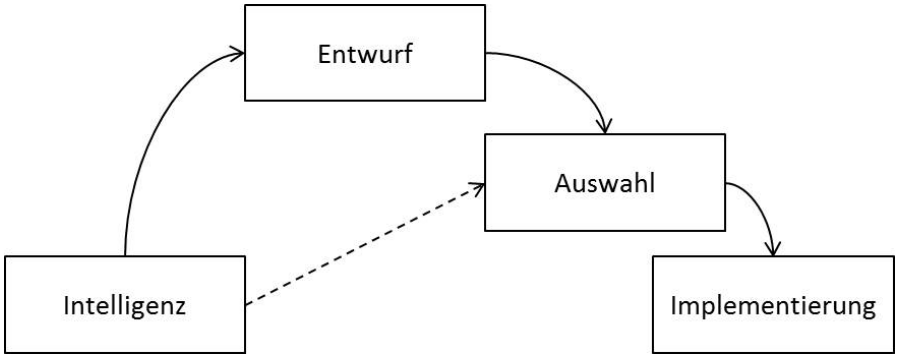


Abbildung 2.13: 4 Phasen der Entscheidungsfindung

unstrukturierten Anforderungen der Abstraktionshierarchie erreicht werden kann?

Der Prozess zur Entwicklung, Bewertung und Auswahl von Entscheidungen soll im Folgenden als *Entscheidungsprozess* modelliert werden (Gasser, Fischer und Wäfler, 2011). Zu diesem Zweck soll eine verkürzte Variante der Entscheidungsleiter von Rasmussen (1986) verwendet werden (vgl. Durso u. a. (2007)). Sie enthält die drei klassischen Entscheidungsphasen Intelligenz, Entwurf und Auswahl (Turban, Sharda und Delen, 2011), die auf Gorry und Morton (1971) zurückgehen und wird in dieser Arbeit durch eine optionale Implementierungsphase ergänzt (Abbildung 2.13). Die einzelnen Phasen können in Bezug auf die wissensbasierte Planung wie folgt interpretiert werden:

Intelligenz: Die Intelligenzphase ist der Einstieg in den Entscheidungsprozess. In einer bestimmten Situation (vgl. Abschnitt 2.4.1) stellt der Disponent einen Handlungsbedarf fest, d.h. die Notwendigkeit, die Eigenschaften des Plans zu beeinflussen. Die Situation kann unmittelbar vor der Planerstellung, unmittelbar nach der Planerstellung (aufgrund des automatisch erzeugten Plans) oder während der Planausführung (aufgrund eines unvorhergesehenen Ereignisses) eintreten. In der Intelligenzphase sammelt der Disponent Wissen, welches zunächst in Form von unstrukturierten Anforderungen vorliegt.

Entwurf: In dieser Phase werden Entscheidungen gesucht, mit denen sich alle strukturierten und unstrukturierten Anforderungen erfüllen lassen. Dabei werden Alternativen entworfen, bewertet und miteinander verglichen.

Auswahl: Die endgültige Entscheidung für eine bestimmte Alternative wird in

der Auswahlphase gefällt, die den Entscheidungsvorgang abschließt. In dieser Phase werden die Entscheidungen bestimmt, die alle strukturierten und unstrukturierten Anforderungen am besten erfüllen.

Implementierung: Die ermittelten Entscheidungen müssen, wenn dies noch nicht in der Entwurfsphase erfolgt ist, in der Lösung umgesetzt bzw. dem Planungsmodell hinzugefügt werden. Im letzteren Fall kann anschließend mit einem regelbasierten Verfahren eine vollständige Lösung erzeugt werden.

Wie in Abbildung 2.13 gezeigt, kann der Entwurfsprozess übersprungen werden, wenn unstrukturierte Anforderungen direkt in Planungsentscheidungen überführt werden können.

Merkmale der Entwurfsphase des Entscheidungsprozesses

Eine in der Entwurfsphase betrachtete Alternative kann sich aus einer, aber auch aus mehreren Planungsentscheidungen für einzelne Aufgaben zusammensetzen. Jede Alternative kann damit als Plan zur Lösung eines Problems (Wie kann die Anforderung umgesetzt werden?) betrachtet werden. Die Suche nach Alternativen soll im Folgenden mit allgemeinen wissenschaftlichen Erkenntnissen über menschliches Problemlösen charakterisiert werden (Fischer, Greiff und Funke, 2012; Bainbridge, 1997; Stacey und Eckert, 2010; Kazakci und Tsoukiàs, 2003).
In der Entwurfsphase eines Entscheidungsprozesses können zunächst Ideen für alternative Problemlösungsmaßnahmen entwickelt werden. Beispiel 2.19 zeigt dies anhand eines Entscheidungsprozesses, der unmittelbar nach der automatischen Planerstellung stattfindet.

Beispiel 2.19: Ein Disponent betrachtet die vom Computer vorgeschlagenen Tagestouren für die Außendienstmitarbeiter seines Unternehmens (Intelligenzphase). Da er mit den Mitarbeitern im Kontakt steht, kann er den Arbeitsaufwand der Kundenbesuche einschätzen. Um Stress zu vermeiden, möchte er erreichen, dass an einem Tag arbeitsaufwändige und weniger arbeitsaufwändige Termine einander ausgleichen (unstrukturierte Anforderung). Der Disponent möchte daher einige Termine aus einer bestimmten Tour entfernen und in eine andere Tagestour einordnen. An diesem Punkt ist er sich aber noch nicht sicher, welche konkrete Maßnahme er ergreifen soll (Entwurfsphase). Er sieht zunächst drei Möglichkei-

ten:

a) einige Aufträge werden einen Tag eher vom gleichen Mitarbeiter ausgeführt,

b) einige Aufträge werden an späteren Tagen von diesem oder anderen Mitarbeitern ausgeführt,

c) die Aufträge werden am gleichen Tag von anderen Mitarbeitern ausgeführt.

Außerdem weiß er noch nicht genau, welche Aufträge aus der Tour entfernt werden sollen.

In der Regel besitzen Menschen nicht genügend Wissen über das Problem, um sofort einen verbindlichen Plan zur Umsetzung einer Problemlösungsidee aufstellen zu können. So haben Disponenten im Gegensatz zum Computer in der Regel nicht das ganze Planungsmodell im Kopf und können daher die benötigten Planungsentscheidungen nicht unmittelbar aus einer Idee ableiten.
Die Aufstellung eines Plans zur Problemlösung erfolgt stattdessen Schritt für Schritt und führt zu Erkenntnissen über das Problem und den Handlungsspielraum. Erst während der schrittweisen Umsetzung einer konkreten Maßnahme durch Planungsentscheidungen erweitert der Disponent sein Expertenwissen über das betrachtete Planungsproblem. Dabei muss er sich zum einen mit den existierenden Randbedingungen auf der Dienst- und Ressourcenebene der Abstraktionshierarchie auseinandersetzen, die nicht verletzt werden dürfen. Um alle Randbedingungen zu erfüllen, muss er ggf. eine Kompromisslösung mit geringfügigen Abweichungen von den beabsichtigten Entscheidungen finden. Zum anderen muss er nach jeder Planungsentscheidung die Auswirkungen auf die Qualität des Plans beurteilen, d.h. er muss überprüfen, ob alle weiteren vorliegenden Anforderungen ausreichend berücksichtigt werden (Cegarra, 2008).

Beispiel 2.20 (Fortsetzung von 2.19): Der Disponent versucht zunächst Möglichkeit a) umzusetzen. Er versucht, einzelne Aufträge einen Tag eher einzuplanen. Er stellt jedoch fest, dass aufgrund von Terminvorgaben kein Auftrag mit einem hohen Arbeitsaufwand einen Tag nach vorn verlegt werden darf. Daraufhin versucht er, einzelne Aufträge einen Tag später einzuplanen und betrachtet mögliche Touren für diesen Tag. Da er feststellt, dass die Abfolge der Touren in Bezug auf

die Fahrzeit und in Bezug auf seine persönlichen Erfahrungen sehr ungünstig ist, verwirft er die die Möglichkeiten a) und b) und entschließt sich, Möglichkeit c) umzusetzen.

Der Disponent muss Maßnahmen verwerfen, die sich nicht umsetzen lassen oder die negative Auswirkungen auf den Gesamtplan haben. Die Auseinandersetzung mit dem Problem kann jedoch gleichzeitig zu neuen Präferenzen und damit zu Ansätzen für eine veränderte Umsetzung einer Maßnahme oder zu einer neuen Maßnahme führen.

Beispiel 2.21 (Fortsetzung von 2.20): Der Disponent möchte nun einige Aufträge auf zwei andere in Frage kommende Mitarbeiter X und Y verteilen, deren Touren nur Aufgaben mit einem geringen Arbeitsaufwand enthalten.

1. Er wählt zwei Aufträge mit einem hohen Arbeitsaufwand für die Umverteilung aus. Aufgrund einer unstrukturierten Anforderung möchte er für sie eine Einplanung am Vormittag vermeiden.

2. Er versucht, den ersten Auftrag bei Mitarbeiter X einzuplanen. Dabei kommt es zu Konflikten mit Aufträgen, die aufgrund ihrer Zeitfenstereinschränkungen ebenfalls am Nachmittag ausgeführt werden müssen. Eine Einplanung bei Mitarbeiter Y ist jedoch möglich.

3. Der zweite Auftrag kann ohne Probleme bei Mitarbeiter X eingeplant werden.

Da die Umplanung erfolgreich ausgeführt werden konnte, verzichtet der Disponent darauf, weitere Szenarien durchzuspielen und andere Aufträge mit einem hohen Arbeitsaufwand für die Umverteilung auszuwählen.

Es wird deutlich, dass der Entwurf und die Auswahl von alternativen Problemlösungsmaßnahmen ineinander verschachtelt auftreten können (Higgins, 1998). Oft ist die Umsetzung einer Maßnahme ein eigenständiger Entscheidungsprozess. Im obigen Beispiel gestaltet sich die Hierarchie der Entscheidungsprozesse wie folgt:

1. Es gibt mehrere mögliche Maßnahmen (Beispiel 2.19).

2. Bei Maßnahme c) stehen mehrere mögliche Ressourcen zur Auswahl. Davon

wird zunächst eine in Frage kommende Ressourcengruppe (zwei Ressourcen) bestimmt (Beispiel 2.21).

3. Bei der jeweils ausgewählten Ressource müssen mehrere mögliche Reihenfolgeentscheidungen erwogen werden (Beispiel 2.21).

2.5 Zusammenfassung und Schlussfolgerungen

Kombinatorische Planungsprobleme besitzen eine gemeinsame Modellstruktur (Abschnitt 2.2). Die einzelnen Ausprägungen der Modellstruktur für unterschiedliche Anwendungsbereiche unterscheiden sich hinsichtlich ihrer Komplexität (Abschnitt 2.3.2). Eine Vielzahl verschiedener Verfahren wurde entwickelt, um auch Probleme mit hoher Komplexität in akzeptabler Zeit lösen zu können (Abschnitt 2.3). Dabei müssen Kompromisse zwischen benötigter Rechenzeit, Lösungsqualität und Genauigkeit des Planungsmodells gefunden werden. Fehlende strukturierte Anforderungen und Ungenauigkeiten im Planungsmodell müssen durch Planungsentscheidungen des Disponenten kompensiert werden (Abschnitt 2.4.1). Diese werden im Rahmen eines kognitiven Entscheidungsprozesses ermittelt (Abschnitt 2.4.3).
Es kann festgestellt werden, dass das Vorgehen zur Ermittlung von Planungsentscheidungen in der Entwurfsphase des Entscheidungsprozesses Gemeinsamkeiten mit den Grundprinzipien von Lösungsverfahren für kombinatorische Planungsprobleme besitzt. So kann, vergleichbar mit den Algorithmen 1 und 3, auf der Grundlage von Gleichung 2.20 die Entwurfsphase als schrittweise Variablen- und Wertauswahl zur Konstruktion einer Lösungsalternative aufgefasst werden. Welche Variablen ausgewählt und wie die Wertebereiche eingeschränkt werden, hängt von den Vorstellungen des Disponenten über die (Weiter-)entwicklung der Alternative und von dem durch die Abstraktionshierarchie der Arbeitsumgebung definierten Handlungsspielraum ab.
Der Gesamtprozess zur Lösung von Planungsproblemen, in den sowohl Mensch als auch Computer integriert sind, kann also formal mit den gleichen Mitteln wie ein Lösungsverfahren für CSPs beschrieben werden. Um die Forschungsfrage zu beantworten, bietet sich daher folgende Vorgehensweise an:

1. Modellierung des integrierten Lösungsprozesses für Planungsprobleme, die der allgemeinen Modellstruktur entsprechen.

2. Identifizierung von Teilen des Lösungsprozesses, die für den Disponenten den Aufwand der Entwurfsphase erhöhen.

3. Konzipierung von Werkzeugen der interaktiven Optimierung, die den Disponenten im Lösungsprozess unterstützen.

Die Umsetzung dieser drei Schritte wird im nächsten Kapitel beschrieben.

3 Gestaltung der Funktionsaufteilung zur interaktiven Lösung von Planungsproblemen

In diesem Kapitel wird ein Konzept der Funktionsaufteilung für kombinatorische Planungsprobleme entwickelt, welches die in Abschnitt 1.3.4 beschriebenen Defizite beseitigt. Zunächst wird der Ablauf der Interaktion zwischen Mensch und Computer als integrierter Lösungsprozess modelliert (Abschnitt 3.1) und hinsichtlich der Qualität der Entscheidungsunterstützung bewertet (Abschnitt 3.2). In Abschnitt 3.3 wird untersucht, bis zu welchem Grad ein existierendes Modell der Funktionsaufteilung für Konfigurationsprobleme zur Verbesserung der Interaktion angewendet werden kann. In den Abschnitten 3.4 bis 3.8 erfolgt eine Spezialisierung und Ergänzung des Modells für Planungsprobleme durch 5 Planungswerkzeuge. Dabei wird sowohl die Implementierung, als auch die Konfiguration und Anwendung der Werkzeuge im integrierten Lösungsprozess diskutiert. In Abschnitt 3.9 werden die Empfehlungen zur Gestaltung der Funktionsaufteilung zu 9 Interaktionsrichtlinien zusammengefasst.

3.1 Modellierung des integrierten Lösungsprozesses für semi-strukturierte Planungsprobleme

Um im Rahmen einer wissensbasierten Planung unstrukturierte Anforderungen umzusetzen, muss das Planungsmodell oder die Lösung modifiziert werden (Abschnitt 2.4.3). An der Benutzeroberfläche werden dazu Werkzeuge der interaktiven

Optimierung benötigt (Abschnitt 1.3.2, Abbildung 1.7). Die Funktionsaufteilung zwischen Mensch und Computer und die damit verbundene wahrgenommene Nützlichkeit wird davon bestimmt, welche Werkzeuge zur Verfügung stehen.
Um geeignete Werkzeuge der interaktiven Optimierung zu finden, soll im Folgenden zunächst ein Lösungsschema der interaktiven und iterativen Planung entwickelt werden. Es beschreibt den menschlichen Entscheidungsprozess zur wissensbasierten Planung unter der Voraussetzung, dass Interaktionswerkzeuge zur Verfügung stehen, die minimale Anforderungen erfüllen und in modernen Planungssystemen bereits häufig angeboten werden (Abschnitt 1.3.3). Um die Beschreibung möglichst formal zu gestalten und um Parallelen zwischen der menschlichen und algorithmischen Vorgehensweise aufzuzeigen, orientiert sich das Schema am Verfahren der Tiefensuche mit Backtracking für CSPs.

3.1.1 Mindestanforderungen an das Planungssystem

Im Folgenden wird vorausgesetzt, dass ein Planungssystem die Möglichkeit bietet, vollständige und ggf. nach bestimmten Zielfunktionen optimierte Pläne auf Knopfdruck zu erzeugen. Die Pläne können dabei Teilpläne für bestimmte Unternehmensabteilungen oder bestimmte Aggregationsstufen sein. So ist z.B. eine Unterteilung in einen Grobplan für das gesamte Unternehmen und Feinpläne für jede einzelne Abteilung denkbar.
Aufgrund der hohen Komplexität werden in der Praxis häufig auch lokal optimale Pläne akzeptiert und die Verletzung von (z.T. weichen) Randbedingungen wird in Kauf genommen. Es wird vorausgesetzt, dass an der Benutzeroberfläche die maximale Rechenzeit eingestellt werden kann, nach welcher ein Such- bzw. Optimierungsvorgang abgebrochen wird.
Der Nutzer sollte die Wertzuweisung für beliebige Variablen des Planungsmodells manuell beeinflussen können. Dazu muss zum einen *vor* der automatischen Planung eine Anpassung des Modells gemäß Gleichung 2.20 ermöglicht werden. Der Disponent sollte unäre Randbedingungen festlegen können, die den Wertebereich ausgewählter Variablen auf genau einen Wert einschränken. Zum anderen muss das Planungssystem eine *nachträgliche* Modifikation einer bestehenden Lösung durch die Anpassung von Wertzuweisungen für beliebige Variablen erlauben. Üblicherweise wird vor jeder manuellen Planänderung durch den Nutzer der aktuelle Zustand des Plans vom Computer abgespeichert, sodass der Disponent später bei

Bedarf wieder darauf zugreifen kann (z.B. durch Aufruf der Aktion „Rückgängig“ bzw. „Wiederherstellen“).
Das Planungssystem muss nach jeder manuell getroffenen Planungsentscheidung die Gültigkeit der Lösung überprüfen und den Nutzer ggf. über eine Verletzung von Randbedingungen benachrichtigen.
Für jede Lösung muss der Disponent eine Übersicht über die Qualität in Form von Leistungskennzahlen erhalten. Diese entsprechen in der Regel den Zielfunktionswerten des Optimierungsmodells. Auch für partielle Lösungen sollten die Kennzahlen dargestellt werden, denn sie erleichtern es dem Disponenten, bei der manuellen Planung die Einhaltung von Qualitätsanforderungen sicherzustellen.
Darüber hinaus kann der Lösungsprozess noch flexibler gestaltet werden, wenn die Möglichkeit zur automatischen Erzeugung partieller Lösungen für eine beliebige Teilmenge von Variablen existiert.

3.1.2 Vorüberlegungen

Die Untersuchung von Entscheidungsprozessen während der Planung (Abschnitt 2.4.3) führte zu der Erkenntnis, dass Planungsentscheidungen häufig nicht ad hoc getroffen werden. Stattdessen finden iterative Prozesse, bei denen Alternativen verglichen werden, und konstruktive Prozesse zur Beschreibung einer einzelnen Alternative statt (vgl. auch Abschnitt 1.4.1).
Im konstruktiven Prozess wird eine partielle Beschreibung der Lösungsalternative schrittweise um Planungsentscheidungen erweitert, wobei der Disponent nach jedem Schritt Rückschlüsse ziehen und Präferenzen entwickeln kann. Dieser Vorgang soll im Folgenden als Suchverfahren auf der Basis eines CSPs modelliert werden. Das Grundprinzip einer schrittweisen Variablen- und Wertauswahl wird von Algorithmus 3 übernommen. Im Gegensatz dazu ist die manuelle Lösungskonstruktion jedoch kein vollständiger und systematischer Suchprozess. Es können folgende Abweichungen identifiziert werden:

Variablenauswahl: Die Variablenauswahl erfolgt nach einer nutzerspezifischen Heuristik. Da es sich um einen Prozess zur Problemlösung handelt, ist die Reihenfolge der Variablenauswahl in der Regel nicht vorbestimmt, sondern sie wird vom Disponenten aufgrund des Zustandes der aktuellen (partiellen) Lösung vorgenommen. Im Unterschied zur Tiefensuche können auch bereits

zugewiesene Variablen, die Teil der aktuellen Lösung sind, erneut für eine Wertzuweisung ausgewählt werden, wenn sich der Disponent während der Suche neues Wissen angeeignet oder neue Präferenzen entwickelt hat.

Wertauswahl: In der Regel betrachtet der Disponent nur einen Teil des Wertebereiches einer Variablen, der für ihn bei der Umsetzung einer unstrukturierten Anforderung interessant ist. Die Wertauswahl erfolgt also ebenfalls aufgrund einer nutzerspezifischen Heuristik.

Backtracking: Wenn die Erweiterung einer partiellen Lösung nicht mehr möglich ist, ohne Randbedingungen zu verletzen („Sackgasse"), so kann der Disponent versuchen, rückwirkend einige seiner Entscheidungen zu ändern. Dieser Vorgang wird im Folgenden als *manuelles Backtracking* bezeichnet. Im Gegensatz zur Tiefensuche wird das manuelle Backtracking jedoch nicht systematisch, wie z.B. als chronologisches Backtracking (Algorithmus 3) durchgeführt. Bei Menschen ist die Kapazität des Arbeitsgedächtnisses stark begrenzt und wird bei Problemlösungsaufgaben nicht zur Speicherung von Pfaden im Suchbaum eingesetzt (vgl. auch Langley und Rogers (1958) und Jones und Langley (2005)). Daher achtet der Disponent beim manuellen Backtracking in der Regel nicht auf die ursprüngliche Zuweisungsreihenfolge. Außerdem gibt es keine vorgegebene Abbruchbedingung, denn der Disponent kann jede Zuweisung beliebig oft ändern.

Der manuelle Suchprozess erfolgt solange, bis der Disponent alle Anforderungen spezifiziert hat. Anschließend kann der Disponent die Lösung vom Computer automatisch vervollständigen lassen, wenn einige Variablen noch keinen Wert besitzen. Dies kann aus folgenden Gründen der Fall sein:

- *Ohne Ausgangslösung:* Der Disponent besitzt nicht für alle Variablen eine Präferenz und hat daher nur eine partielle Lösungsalternative erzeugt.
- *Mit Ausgangslösung:* Der Disponent hat einige Zuweisungen in der automatisch erzeugten Ausgangslösung verworfen, da es ihm anderweitig nicht gelungen ist, die ihn interessierenden Variablen ohne Verletzung von Randbedingungen mit Werten zu belegen.

Nach der automatischen Planungsphase kann der Disponent erneut eine manuelle Anpassung vornehmen, wenn aus seiner Sicht noch nicht alle unstrukturierten

Anforderungen erfüllt sind. Eine Anpassung ist zudem auch dann erforderlich, wenn der Computer keine Lösung unter Einbeziehung der manuellen Zuweisungen finden konnte. Der Disponent muss frei entscheiden können, wie oft sich manuelle und automatische Planung bei der Erstellung einer Lösungsalternative abwechseln.

3.1.3 Ein interaktiver Algorithmus

Teil 1 - der übergeordnete Entscheidungsprozess

Algorithmus 5 zeigt den vollständigen Ablauf des interaktiven Lösungsprozesses[1]:

Zeilen 2-7: Der Algorithmus sieht vor, dass der Disponent zunächst entscheidet, ob vom Computer eine vollständige Ausgangslösung erzeugt werden soll ($userInitialize = true$ oder $false$). Auf diese Weise können in vereinfachter Form alle typischen Planungssituationen abgebildet werden (Planung mit oder ohne Ausgangslösung sowie die nachträgliche Modifikation eines existierenden Plans). Die Prozedur *computerSolve*() repräsentiert einen beliebigen Lösungs- oder Optimierungsalgorithmus (z.B. nach dem Vorbild von Algorithmus 3). Sie kann vom Nutzer konfiguriert werden (Liste *settings* als Parameter der Prozedur *computerSolve*()). Dies ermöglicht z.B. die Wahl eines Optimierungsverfahrens oder die Gewichtung von Zielfunktionen (vgl. Gleichung 2.3).

Zeilen 9-16: Anschließend sind entweder keine oder alle Variablen zugewiesen (Liste *assigned*). Der Disponent kann sich nun dafür entscheiden, die Prozedur *userSolve*() durchzuführen. Die Entscheidung wird durch die Variable *userAssignOrUnassign* repräsentiert, die den Wert *true* oder $false$ annehmen kann. Die Prozedur beinhaltet, dass der Disponent eine noch nicht zugewiesene Variable (Liste *unassigned*) manuell zuweist oder eine bestehende Zuweisung ändert oder löst (vgl. Algorithmus 6). Anschließend kann er die

[1] **Konventionen für die folgenden Interaktions-Algorithmen:** Prozeduren mit dem Präfix *user* repräsentieren Entscheidungen oder (z.T. rein kognitive) Aktivitäten des Disponenten. Der Präfix *computer* kennzeichnet eine unterstützende Prozedur, die vom Disponenten an der Benutzerschnittstelle aufgerufen werden kann. Im Kopf einer Bedingung (if..then) werden z.T. Variablen mit dem Präfix *user* eingesetzt, deren Wert für eine Ja/Nein-Entscheidung des Disponenten steht.

geänderte Lösung vom Computer weiterverarbeiten lassen (Zeile 14, in diesem Fall sei $userComplete = true$, ansonsten $false$). Der Computer kann z.B. eine partielle Lösung vervollständigen, wobei die in der Liste *assigned* bereits vorliegenden Wertzuweisungen als Randbedingungen berücksichtigt werden[2]. Auch die Wiederherstellung der Ausgangslösung bzw. einer in einer vorangegangenen Iteration erzeugten Alternativlösung durch den Computer ist in diesem Schritt denkbar, wenn der Disponent mit dem Ergebnis seiner Modifikation unzufrieden ist („Rückgängig"-Funktion). Die Weiterverarbeitung wird durch die Prozedur *computerSolve*() (Zeile 15) repräsentiert, wobei der Nutzer die gewünschte Aktion über den Parameter *settings* konfiguriert.

Zeilen 8-17: Die in den Zeilen 9-16 geschilderten Schritte können beliebig oft wiederholt werden, bis der Disponent den Vorgang abbricht ($userCancelled = true$).

Mit Zeile 8-17 wird die Entwurfsphase des Entscheidungsprozesses modelliert. Durch die wiederholte Ausführung der Prozedur *userSolve*() setzt der Disponent unstrukturierte Anforderungen in strukturierte Anforderungen um. Dabei muss er stets die Erfüllung sämtlicher Randbedingungen, Anforderungen und Zielen der Arbeitsumgebung sicherstellen. Hilfsmittel sind dafür die Prozedur *computerSolve*() bzw. die in Abschnitt 3.1.1 beschriebenen Systemfunktionalitäten. In den Iterationen können auf diese Weise mehrere Problemlösungsmaßnahmen evaluiert werden (vgl. Beispiel 2.19). Die letzte Iteration entspricht der Implementierungsphase, in der der Disponent die ausgewählte Maßnahme umsetzt bzw. wiederherstellt. Die dazwischen liegende, gedanklich stattfindende Auswahlphase wird nicht explizit modelliert.

[2] Anschließend befinden sich alle Variablen aus *unassigned* in der Liste *assigned* und besitzen Werte.

Algorithmus 5 : Interaktive Lösung eines CSPs für die Planung (Teil 1)

input : Ein CSP (V, D, C)

output : Eine Lösung $assigned = \{\langle v_1, w_1\rangle, ..., \langle v_n, w_n\rangle\}$ mit $n = |V|$, $\forall\langle v, w\rangle \in assigned : v \in V, w \in dom(v)$, wenn eine Lösung existiert; **false**, wenn keine Lösung existiert.

```
1  begin
2      unassigned ← V;
3      assigned ← ∅;
4      /*Ausgangslösung automatisch erzeugen?*/
5      if userInitialize then
6          computerSolve(assigned, unassigned, settings);
7      end
8      while !userCancelled do
9          /*Zuweisungen manuell vervollständigen, ändern oder lösen:*/
10         if userAssignOrUnassign then
11             userSolve(assigned, unassigned);
12         end
13         if userComplete then
14             /*automatische Vervollständigung oder Anpassung der (partiellen)
                 Lösung:*/
15             computerSolve(assigned, unassigned, settings);
16         end
17     end
18 end
```

Teil 2 - der Teilprozess zur Modifikation einer (partiellen) Lösung

Algorithmus 6 zeigt den Ablauf zur Modifikation einer Alternativlösung durch den Disponenten. Zunächst wird eine beliebige bereits zugewiesene oder noch nicht zugewiesene Variable ausgewählt (Zeile 2). Eine zugewiesene Variable wird vom Disponenten sofort in die Liste noch nicht zugewiesener Variablen verschoben (Zeilen 4-5). Algorithmus 6 ist in diesem Fall abgeschlossen und der Disponent kann sich anschließend in Algorithmus 5 (Zeile 13) entscheiden, ob die Wertzuweisung für diese Variable in der nächsten Iteration manuell ($userComplete = false$) oder in

dieser Iteration vom Computer ($userComplete = true$) vorgenommen werden soll. Wenn die Variable in *unassigned* vorliegt, wird für sie in den Zeilen 7 bis 18 ein neuer Wert gesucht. Der Prozess gestaltet sich wie folgt:

Zeile 7: Die ausgewählte Variable wird zunächst aus der Liste *unassigned* entfernt.

Zeile 8: Der Disponent wählt eine Reihe von alternativen Werten des Wertebereiches aus, die aus seiner Sicht für eine Zuweisung in Frage kommen.

Zeile 10: Alle ausgewählten Werte werden vom Disponenten daraufhin überprüft, ob sie innerhalb der aktuellen partiellen Lösung zulässig und zufriedenstellend sind. Ein Wert ist zulässig, wenn seine Zuweisung keine Randbedingungen mit bereits zugewiesenen Variablen verletzt. Der Disponent wird bei der Ermittlung der Zulässigkeit vom Computer unterstützt (vgl. Abschnitt 3.1.1). Ein Wert ist zufriedenstellend, wenn die um diese Wertzuweisung ergänzte (partielle) Lösung aus Sicht des Disponenten eine ausreichende Qualität hinsichtlich der Erfüllung strukturierter und unstrukturierter Anforderungen besitzt. Die Prozedur *userSelectCandidates* repräsentiert diesen Auswahlvorgang und gibt alle zulässigen und zufriedenstellenden Werte zurück.

Zeile 12: Wenn kein passender Wert existiert, wird die ausgewählte Variable wieder in die Liste *unassigned* verschoben.

Zeilen 14-15: Wenn passende Werte existieren, wählt der Nutzer einen Wert aus (Zeile 14). Dieser wird der ausgewählten Variable zugewiesen (Zeile 14). Die Variable wird dann in die Liste *assigned* verschoben (Zeile 15).

Zeile 18: Der Teilprozess zum Zurücknehmen oder Ändern einer Zuweisung ist nun abgeschlossen.

Algorithmus 6 : Interaktive Lösung eines CSPs für die Planung (Teil 2)

```
1  userSolve(assigned, unassigned) ≡
2  /*Variablenauswahl aus assigned oder unassigned:*/
     v_select ← userSelectVar(assigned, unassigned);
3  if ∃⟨v, w⟩ ∈ assigned : v = v_select, w ∈ dom(v) then
4  |   assigned ← assigned\{⟨v_select, w⟩};
5  |   unassigned ← unassigned ∪ {v_select};
6  else
7  |   unassigned ← unassigned\{v_select};
8  |   subDomain ← userSelectValues(D_select);
9  |   candidates ← ∅;
10 |   /*Wahl von Kandidaten, die zulässig und zufriedenstellend sind:*/
   |     candidates ← userSelectCandidates(subDomain);
11 |   if candidates = ∅ then
12 |   |   unassigned ← unassigned ∪ {v_select};
13 |   else
14 |   |   value ← userSelectValue(candidates);
15 |   |   assigned ← assigned ∪ {⟨v_select, value⟩}};
16 |   end
17 end
18 return;
```

Die Wertzuweisung einer ausgewählten Variable kann als eingebetteter Entscheidungsprozess mit einer Entwurfsphase (Zeile 10), Auswahlphase (Zeile 14) und Implementierungsphase (Zeile 15) betrachtet werden.

Beispiel 3.1: Die Fallstudie der Beispiele 2.19 bis 2.21 soll im Folgenden beispielhaft auf das in den Algorithmen 5 und 6 gezeigte Lösungsschema abgebildet werden. Dabei wird ein VRPTW-Modell zugrunde gelegt wird (Abschnitt 2.2.3).
Algorithmus 5:

- Ausgangssituation ist ein automatisch erzeugter Plan, d.h. es gilt $userInitialize = true$ (Zeile 5).
- Da der Disponent mit dem Plan noch nicht zufrieden ist, gilt $userCancelled = false$ (Zeile 8).
- Der Disponent möchte nun einige Aufgaben manuell um einen Tag nach

vorn verlegen (vgl. Beispiel 2.19). Daher gilt $userAssignOrUnassign = true$ (Zeile 10).

- **Algorithmus 6** - Eintritt in die Prozedur $userSolve()$
- Der Disponent möchte eine Aufgabe a_{select} zeitlich verschieben. Daraus ergibt sich:
- $v_{select} = a_{select_{start}}$ (Zeile 2).
- v_{select} ist in der Liste $assigned$ enthalten (Zeile 3),
- Nach Ausstieg aus Algorithmus 6 (Zeile 18) weiter in Algorithmus 5 (Zeile 13) mit $userComplete = false$ und $userCancelled = false$. In der nächsten Iteration gilt $userAssignOrUnassign = true$, Wiedereinstieg in Algorithmus 6. Nun werden die Anweisungen ab Zeile 7 betrachtet.
- Der Wert von $subDomain$ umfasst die Arbeitszeit des Tages, dem a_{select} zugeordnet werden soll (Zeile 8).
- Alle Zeitpunkte in $subDomain$ sind unzulässig, da sie zu einer Verletzung der für a_{select} vorliegenden Zeitfensterbedingung führen (Zeile 10, vgl. Beispiel 2.20 und Gleichung 2.14).
- Daher gilt: $candidates = \emptyset$ (Zeile 11) und Algorithmus 6 wird ohne Wertzuweisung für v_{select} beendet.

- **Algorithmus 5:** Der Disponent stellt die ursprüngliche Startzeit für a_{select} wieder her (z.B. über eine „Rückgängig“-Funktion). Daher gilt $userComplete = true$ (Zeile 13). Er versucht in den nächsten Iterationen, weitere Aufgaben einen Tag früher oder später einzuplanen. Dies ist ohne Erfolg, da Zeitfensterbedingungen verletzt werden oder Optimierungsziele nicht erreicht werden (vgl. Beispiel 2.20).
- Der Disponent möchte nun die Mitarbeiterzuordnung für einige Aufgaben ändern (vgl. Beispiel 2.21). Daher gilt weiterhin $userCancelled = false$ (Zeile 8).

- Es gilt $userAssignOrUnassign = true$ (Zeile 10).
  - **Algorithmus 6** - Eintritt in die Prozedur $userSolve()$
  - Der Disponent möchte eine Aufgabe a_{select} den Ressourcen res_x oder res_y zuordnen. Daraus ergibt sich:
  - $v_{select} = a_{select_{res}}$ (Zeile 2),
  - v_{select} ist in der Liste $assigned$ enthalten (Zeile 3),
  - Nach Ausstieg aus Algorithmus 6 (Zeile 18) weiter in Algorithmus 5 (Zeile 13) mit $userComplete = false$ und $userCancelled = false$. In der nächsten Iteration gilt $userAssignOrUnassign = true$, Wiedereinstieg in Algorithmus 6. Nun werden die Anweisungen ab Zeile 7 betrachtet.
  - $subDomain = \{res_x, res_y\}$ (Zeile 8),
  - Zunächst sind beide Zuweisungen zulässig und zufriedenstellend (Zeile 10).
  - Der Disponent wählt res_x aus (Zeile 14), sodass gilt: $a_{select_{res}} = res_x$ (Zeile 15). Algorithmus 6 wird beendet.
- **Algorithmus 5:** Die Einplanung der Aufgabe a_{select} in die Tour von Mitarbeiter X ist jedoch erst vollständig, wenn auch für $a_{select_{start}}$ ein Wert ermittelt wurde. Dies könnte im nächsten Schritt automatisch über die Prozedur $computerSolve()$ (Zeile 15) erfolgen. Da der Disponent jedoch eine Einplanung am Nachmittag erreichen will, gilt $userComplete = false$. Außerdem gilt $userCancelled = false$.
- In der nächsten Iteration gilt $userAssignOrUnassign = true$ (Zeile 10).
  - **Algorithmus 6** - Eintritt in die Prozedur $userSolve()$
  - $v_{select} = a_{select_{start}}$ (Zeile 2),
  - Der Wert von $subDomain$ umfasst die Arbeitszeit des Tages, dem a_{select} zugeordnet werden soll (Zeile 8).

  - Der Disponent erprobt nun einzelne Zeitpunkte für $a_{select_{start}}$ (Zeile 10).
  - In der bestehenden Tour dieses Tages gibt es keine Lücke, in die Aufgabe a_{select} eingeplant werden könnte. Jeder Zeitpunkt in *subDomain* führt zu einer Verletzung der Randbedingung in Gleichung 2.15 (Überlappungsfreiheit). Daher gilt $candidates = \emptyset$ (Zeile 11) und Algorithmus 6 wird ohne Wertzuweisung beendet.

- **Algorithmus 5 - Zusammenfassung der weiteren Aktionen:** Der Disponent versucht nun in weiteren Iterationen, die anderen in der Tour von res_x befindlichen Aufgaben zeitlich zu verschieben, um eine Lücke für Aufgabe a_{select} zu schaffen. Es gelingt ihm jedoch nicht, eine Überlappung von Aufgaben oder die Verletzung von Zeitfenster-Randbedingungen innerhalb der Tour von res_x zu vermeiden. Nach einigen Iterationen ändert er daher die Ressourcenzuweisung für $a_{select_{start}}$ von res_x zu res_y. In der Tour von res_y sind passende zeitliche Lücken vorhanden, woraus mehrere kandidierende Zeitpunkte am Nachmittag resultieren. Der Disponent wählt daraus einen Zeitpunkt aus. Anschließend setzt er den Lösungsprozess mit einer weiteren Aufgabe $a_{select2}$ fort. Hier gelingt eine Ressourcen- und Startzeitzuweisung bei Mitarbeiter X (res_x).

3.2 Bewertung des Lösungsprozesses hinsichtlich der zu erwartenden Nutzerakzeptanz

Das Lösungsschema in Algorithmus 6 zeigt, welche Aktivitäten zur Lösung semistrukturierter Planungsprobleme erforderlich sind. Ein Lösungsprozess nach diesem Vorbild weist jedoch Defizite auf, welche die Nutzerakzeptanz, insbesondere die wahrgenommene Nützlichkeit des Planungssystems beeinträchtigen können. Sie sind vom Interaktionsstil des Anwenders, vom gewählten Optimierungsverfahren und von den Eigenschaften des Planungsmodells abhängig. Angelehnt an die Feststellungen in Abschnitt 1.3.4 werden Folgenden die Defizite entschlüsselt. Zunächst werden allgemeine Faktoren identifiziert, welche die Nutzerakzeptanz für den Lösungsprozess beeinträchtigen können, da sie einen erhöhten Arbeitsaufwand

verursachen. Anschließend werden in Abschnitt 3.2.2 unterschiedliche Interaktionsmodelle aus dem Lösungsschema abgeleitet. Sie werden im letzten Abschnitt hinsichtlich der zu erwartenden Nutzerakzeptanz untersucht.

3.2.1 Allgemeine Faktoren, die den Aufwand der wissensbasierten Planung beeinflussen

Im Folgenden sollen Faktoren untersucht werden, die den *Arbeitsaufwand* des Disponenten bei der wissensbasierten Planung beeinflussen. Dieser ist ein wichtiges Kriterium für die zu erwartende Nutzerakzeptanz, insbesondere der Nützlichkeit: Je höher dieser Aufwand für den Disponenten, desto geringer ist die zu erwartende wahrgenommene Nützlichkeit.

Aufwand, der durch das automatische Planungsverfahren verursacht wird

Der Aufwand der wissensbasierten Planung wird zum Teil durch die vom Computer benötigte Rechenzeit für die automatische Planung oder Vervollständigung von Plänen bestimmt (Algorithmus 5, Zeilen 6 und 15). Die Rechenzeit sollte im Idealfall nur wenige Sekunden betragen, da die automatische Planung während des Entscheidungsprozesses u.U. mehrmals aufgerufen wird und somit den Entscheidungsprozess mehrmals um Wartezeiten verlängert. Der Einsatz von schnellen, heuristischen Algorithmen (Abschnitt 2.3.4, 2.3.5) ist in diesem Fall einem exakten Optimierungsverfahren mit deutlich höherer Laufzeit vorzuziehen. Die Umsetzung unstrukturierter Anforderungen wird dadurch dem Disponenten erleichtert, da er in kürzerer Zeit mehr Alternativen auswerten kann und somit schneller eine zufriedenstellende Lösung erhält. Qualitätsverluste können bzw. müssen dafür in der Regel in Kauf genommen werden. Die Existenz eines solchen Algorithmus wird im restlichen Teil von Abschnitt 3.2 vorausgesetzt[3].

[3] Wenn ein solcher Algorithmus nicht zur Verfügung steht, sollte die Suche nach einer im System konfigurierbaren Timeoutzeit automatisch abgebrochen werden. Der Anwender sollte in diesem Fall vom System darüber informiert werden, dass innerhalb der vorgegebenen Rechenzeit mit den vorliegenden Randbedingungen kein Plan gefunden werden konnte. Die zuletzt betrachtete Lösung sollte in diesem Fall wiederhergestellt werden.

Aufwand, der durch manuelles Backtracking verursacht wird

Herkömmliche Planungssysteme ermöglichen zwar i.d.R. manuelle Entscheidungen, unterstützen aber häufig nicht deren Propagierung. Für den Disponenten ist es daher nicht ohne Weiteres feststellbar, ob eine manuelle Wertzuweisung *inkonsistent* ist, d.h. dazu führt, dass eine weitere, noch nicht zugewiesene Variable einen leeren Wertebereich besitzt. Er bemerkt den Konflikt erst, wenn das System bei der Zuweisung der betroffenen Variable die Verletzung einer Randbedingung anzeigt. In diesem Fall muss er einen manuellen Backtracking-Vorgang vornehmen, wodurch sich der Aufwand der Suche erhöht.
Die Notwendigkeit für manuelles Backtracking kann sich an verschiedenen Stellen in Algorithmus 5 und 6 ergeben:

- Der Disponent hat eigene Planungsentscheidungen vorgenommen, die in der Liste *assigned* vorliegen (Algorithmus 5, Zeile 11). Bei der automatischen Planung stellt sich jedoch heraus, dass im Lösungsraum keine vollständige Lösung mit dieser Kombination von Eigenschaften existiert (Zeile 15). Der Disponent muss seine Entscheidungen in den nächsten Iterationen abwandeln.

- Der Disponent kann die eigenen Planungsentscheidungen nicht vollständig umsetzen. Ab einem bestimmten Punkt lässt sich die Spezifikation einer Alternativlösung nicht mehr mit weiteren eigenen Planungsentscheidungen fortsetzen, ohne Randbedingungen zu verletzen (es gilt $candidates = \emptyset$ in Algorithmus 6, Zeile 11). Er muss daher die bisher erstellte partielle Lösung abwandeln.

Der Disponent befindet sich in beiden Fällen in einer „Sackgasse“. Er hat in der Regel keine Möglichkeit, das Auftreten von Sackgassen durch eine vorausschauende Planung zu verhindern. Wenn er eine Planungsentscheidung trifft, kann er die Auswirkungen auf die restlichen Variablen nicht sofort feststellen, denn bei einer großen Menge an Variablen und Randbedingungen ist es ihm nahezu unmöglich, im Kopf eine vorausschauende Propagierung durchzuführen. Der Grund ist die begrenzte Gedächtniskapazität des Menschen (MacDonald, 2008), die von der Größe praktischer Planungsprobleme in der Regel weit überschritten wird.
Ob der Wertebereich einer Variablen nach der Propagierung seiner Entscheidung keine Werte mehr enthält (Abschnitt 2.3.6), kann der Disponent also erst dann

feststellen, wenn er die betroffene Variable betrachtet. Diese steht u.U. in der vom Disponenten gewählten Zuweisungsreihenfolge sehr weit hinten. Es finden also u.U. zahlreiche Belegungen nicht betroffener Variablen statt, bevor ein Konflikt bemerkt wird. An dieser Stelle ist die Konfliktursache jedoch meist nicht mehr nachvollziehbar, d.h. der Disponent kann die Variable, für die ein Konflikt vorliegt, nicht mehr der Variable zuordnen, die den Konflikt verursacht hat. Letztere muss, wenn der Wertebereich leer ist oder nicht mehr den gewünschten Wert enthält, in einem zeitaufwändigen manuellen Backtracking-Prozess herausgefunden werden.
Der durch manuelles Backtracking verursachte Aufwand hängt von der Struktur des Planungsmodells ab. Planungsprobleme, die nicht global konsistent sind und eine niedrige Lösungsdichte besitzen, sind aufgrund ihres Lösungsaufwandes nicht für eine manuelle Bearbeitung durch den Disponenten geeignet (vgl. Abschnitt 2.3.2 und 2.3.6). Dies bestätigt auch der Anwendertest für die Fallstudie Tourenplanung (Kapitel 5.3, Metrik Erfolgsrate). Für Planungsprobleme, die global konsistent sind bzw. eine sehr hohe Lösungsdichte besitzen, kann dagegen sehr einfach eine (partielle) Lösung manuell erzeugt werden. In diesem Fall kann der Disponent die Zuweisung mit jeder beliebigen Variable beginnen, in jeder beliebigen Reihenfolge fortsetzen und jeweils beliebige Werte aus den Wertebereichen zuweisen (unter Beachtung der Randbedingungen), ohne in eine Sackgasse zu geraten. Auch eine anschließende automatische Vervollständigung der Lösung ist stets erfolgreich. Abbildung 3.1 zeigt einen Vergleich zwischen zwei Problemen, die jeweils global konsistent und nicht global konsistent sind. Dabei wurden die Beispiele 2.5 und 2.7 als Planungsprobleme interpretiert. Analog zu Beispiel 2.5 kann beim nicht global konsistenten Problem in Abhängigkeit von der Variablenauswahl Backtracking erforderlich sein.

Aufwand, der durch die Optimierung der Zielfunktion verursacht wird

Disponenten bevorzugen einen Plan, bei dem sowohl die unstrukturierten, als auch die strukturierten Anforderungen (z.B. minimale Ausführungskosten und -zeiten) in ausreichendem Maß erfüllt werden. Der Computer ist mit einem heuristischen oder exakten Verfahren in der Lage, für das um eine oder mehrere Randbedingungen ergänzte Planungsmodell eine (annähernd) optimale Lösung zu ermitteln. Eine unstrukturierte Anforderung ist jedoch nicht immer eindeutig einem bestimmten Satz von Randbedingungen zuzuordnen. Oft gibt es aus Sicht des Disponenten

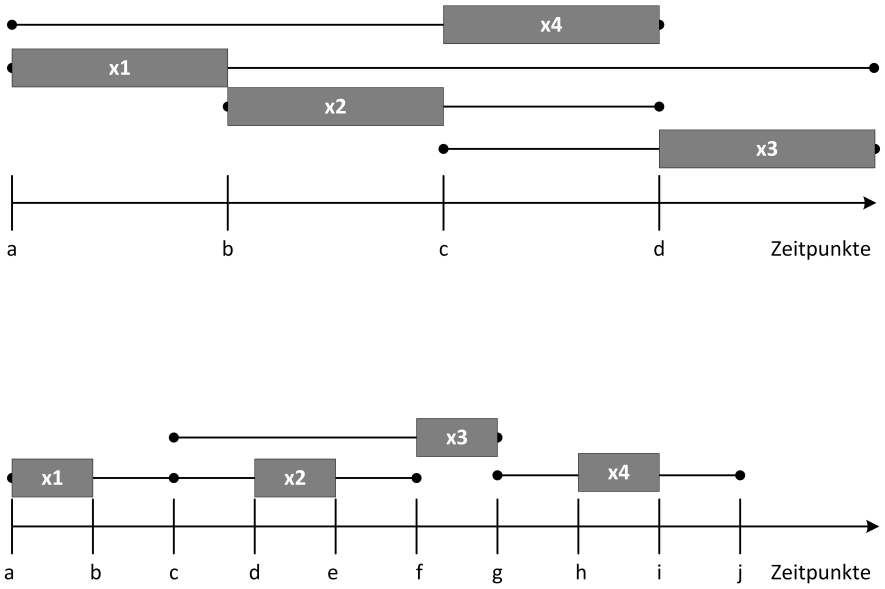


Abbildung 3.1: Ein global konsistentes Planungsproblem (unten) und ein nicht global konsistentes Problem (oben). Die Aufgaben sind durch graue Balken gekennzeichnet, die entlang der Zeitleiste innerhalb ihres Zeitfensters verschoben werden können. Die Aufgaben dürfen sich nicht überlappen.

mehrere Umsetzungsmöglichkeiten, aus denen diejenige gewählt werden muss, mit denen hinsichtlich der strukturierten Anforderungen die beste Lösung erzielt werden kann. Dieser Vorgang erhöht den Aufwand der wissensbasierten Planung.

3.2.2 Interaktionsmodelle

Die Anwendung des in Algorithmus 5 und 6 gezeigten Lösungsschemas kann nach verschiedenen Interaktionsmodellen erfolgen:

- *Manuelle Planung:* Der Disponent nimmt alle Variablenzuweisungen manuell vor, sodass eine Vervollständigung durch die Prozedur $computerSolve()$ nicht mehr nötig ist (in Algorithmus 5 gilt stets: $userInitialize = false$, $userAssignOrUnassign = true$ und $userComplete = false$).

- *Auswahl aus alternativen Lösungen:* Der Disponent lässt stets alle Zuweisungen automatisch vornehmen und unternimmt keine Aktivitäten zur Variablen- und Wertauswahl (in Algorithmus 5 gilt stets:
$userInitialize = true$, $userAssignOrUnassign = false$, $userComplete =$

true). Die Prozedur *computerSolve*() wird beliebig oft mit unterschiedlichen Einstellungen aufgerufen. Alternativ kann das Planungssystem auch eine Menge alternativer Lösungen generieren, die sich auf und in der Nähe der Pareto-Front befinden. Der Disponent muss daraus eine Auswahl treffen (wie z.B. bei Ruiz, Maroto und Alcaraz (2004)).

- *Automatische Planung mit nachträglicher manueller Modifikation:* Der Disponent lässt eine Ausgangslösung automatisch erzeugen ($userInitialize = true$), die von ihm anschließend manuell modifiziert wird (in allen Iterationen gilt $userAssignOrUnassign = true$ und $userComplete = false$).

- *Automatische Planung unter Berücksichtigung zusätzlicher Randbedingungen:* Der Disponent erzeugt keine Ausgangslösung ($userInitialize = false$). Er nimmt einige Zuweisungen manuell vor und lässt die restlichen Zuweisungen vom Computer vervollständigen.

Es existieren weitere Interaktionsmodelle, die eine Mischung bzw. Kombination der hier genannten Modelle sind, wie z.B. eine automatische Planung mit Randbedingungen und nachträglicher Modifikation oder die automatische Erstellung eines partiellen Plans, der manuell vervollständigt wird. Im Folgenden werden nur die hier genannten Basismodelle betrachtet.

3.2.3 Bewertung einzelner Interaktionsmodelle

Im Folgenden werden einzelne Interaktionsmodelle auf Defizite in der Funktionsaufteilung zwischen Mensch und Computer untersucht.

Manuelle Planung

Eine manuelle Planung ist für den Disponenten in der Regel mit einem hohen Aufwand verbunden. Dies gilt besonders dann, wenn das Planungsproblem eine geringe Lösungsdichte besitzt und während der Zuweisung häufig Sackgassen auftreten (vgl. Abschnitt 3.2.1). In Abhängigkeit vom Modell und von der Anzahl der Variablen kann es für den Disponenten zudem schwierig bzw. unmöglich sein, neben der Erfüllung der Randbedingungen auch eine Optimierung der Zielfunktionen zu erreichen.

Bereitstellung alternativer Lösungen durch den Computer

Bei diesem Modell wird die Planerstellung vollständig dem Computer überlassen, aber der Disponent kann in der Entwurfsphase zwischen alternativen Lösungen wählen. Dadurch kann sichergestellt werden, dass der ausgewählte Plan die strukturierten Anforderungen erfüllt und z.B. eine ausreichend gute Qualität besitzt. Außerdem entfällt der Aufwand der manuellen Planung.
Dieser Ansatz kann jedoch zu einer schlechten Nutzerakzeptanz führen, da die Auswertung zahlreicher, einander ähnelnder Pläne sehr ermüdend ist[4]. Er führt zu dem in Abschnitt 1.3.1 beschriebenen Problem der Mensch-Computer-Trennung. Damit verbunden ist ein Kontrollverlust des Disponenten, der keine Möglichkeit erhält, die Eigenschaften der erzeugten Pläne, d.h. die Zuweisungen für einzelne Aufgaben, direkt mitzubestimmen. Der Computer stellt nur eine Teilmenge des Lösungsraums zur Auswahl bereit. Somit steht dem Disponenten nur ein begrenzter Handlungsspielraum zur Verfügung, um unstrukturierte Bedingungen und Anforderungen umzusetzen.

Automatische Planung mit nachträglicher manueller Modifikation

Viele Planungssysteme bieten eine vollautomatische Planerstellung auf Knopfdruck an. Der Disponent erhält die Möglichkeit, den Plan anschließend manuell zu modifizieren (wie z.B. bei Anh, Tam und Hung (2007), vgl. Nascimento und Eades (2005)). Dieses Modell besitzt folgende Nachteile:

Aufwand der manuellen Planung: Bei der manuellen Modifikation besteht die Gefahr, Sackgassen zu erreichen, sodass manuelle Backtracking-Vorgänge notwendig sind. Diese können einen hohen Arbeitsaufwand verursachen.

Verschlechterung der Qualität: Nachträgliche lokale Modifikationen können die Qualität des Plans bezüglich ein oder mehrerer Kennzahlen verschlechtern (Framinan und Ruiz, 2012). Der Disponent sucht in der Regel einen Plan, der die unstrukturierten Anforderungen umsetzt und gleichzeitig die bestmögliche Qualität hinsichtlich der strukturierten Anforderungen besitzt. Er

[4]Sanderson (1989) stellte in einer Studie fest, dass die Auswertung einer großer Anzahl von Plänen eine sehr hohe mentale Beanspruchung darstellt (Cegarra und Wezel (2011b), vgl. auch Burstein u. a. (1996) und Cegarra und Hoc (2008)).

arbeitet dabei nach dem Prinzip einer Verbesserungsheuristik (Abschnitt 2.3.5): Jede manuelle Planänderung, bei der die Randbedingungen berücksichtigt werden, führt zu einer Nachbarschaftslösung mit einer besseren oder schlechteren Qualität. Bei dieser Vorgehensweise besteht die Möglichkeit, ein lokales Optimum zu erreichen. Allerdings befindet sich häufig im Lösungsraum des Planungsproblems ein Plan, der die unstrukturierten Anforderungen ebenfalls erfüllt und gleichzeitig eine bessere Qualität besitzt. Es ist jedoch schwierig, diesen Plan manuell zu finden. In der Regel ist dazu eine größere Umstellung notwendig, die mit einem hohen Arbeitsaufwand verbunden ist.

Complacency-Effekt: Dieser Effekt kann dazu führen, dass nach einer automatischen Planerstellung keine manuellen Modifikationen am Plan vorgenommen werden, obwohl noch nicht erfüllte Präferenzen vorhanden sind (Cegarra und Hoc, 2008). Zum einen weiß der Disponent, dass es ihm (im Gegensatz zum Computer) schwerfällt, gute Qualitätskennzahlen des Gesamtplans zu erreichen. Zum anderen sind ihm u.U. nicht alle Randbedingungen und Planungsregeln bekannt, die in das Planungssystem eingepflegt wurden. Zudem erschwert der hohe Aufwand der manuellen Planung einen Entscheidungsprozess, bei dem mehrere Lösungsalternativen gegeneinander abgewogen werden. Aufgrund der Befürchtung, Fehlentscheidungen zu treffen, erscheint es letztendlich am sichersten, die Planung vollständig dem Computer zu überlassen.

Automatische Planung unter Berücksichtigung zusätzlicher Randbedingungen

Einige Planungssysteme erlauben die Festlegung zusätzlicher Randbedingungen, bevor die automatische Planung gestartet wird (wie z.B. bei Gacias, Cegarra und Lopez (2012), vgl. Nascimento und Eades (2005), Müller und Barták (2002)). Die Optimierung der Zielfunktionen (strukturierte Anforderungen) kann dabei dem Computer überlassen werden. Trotzdem weist das Modell einige Defizite bezüglich der Funktionsaufteilung auf:

Planung durch Versuch und Irrtum: Dem Disponenten wird keine Unterstützung dabei geboten, die Randbedingungen nach dem Gesichtspunkt der Lösbar-

keit auszuwählen (es erfolgt keine Propagierung der Randbedingungen). Die automatische Planung muss u.U. mehrfach mit angepassten Randbedingungen aufgerufen werden, bevor das Problem lösbar ist (mehrere Iterationen in Algorithmus 5, Zeilen 8-17). Dies entspricht einer Erhöhung des Aufwands durch manuelles Backtracking (Abschnitt 3.2.1).

Mangelnde Kontrolle über das Ausmaß der Umplanung: Die nachträgliche Modifikation eines Plans kann auch dadurch realisiert werden, dass der Disponent einige der bestehenden (für seine Entscheidungen nicht relevanten) Zuweisungen zunächst löst, um bestimmte Planänderungen bequemer formulieren zu können (Algorithmus 6, Zeilen 4-5). Nachdem er seine Modifikationen als zusätzliche Randbedingungen eingegeben hat, kann er den Plan automatisch vervollständigen lassen (Algorithmus 5, Zeile 15). Häufig ist es bei einer nachträglichen Änderung jedoch erwünscht, dass die bestehenden Planungsentscheidungen, abgesehen von den Modifikationen des Disponenten, nur in einem bestimmten Umfang verändert werden. Eine automatische Neuplanung modifiziert den Plan jedoch u.U. stärker als erwünscht. Die resultierenden Planänderungen sind u.U. weitreichend und schwer nachvollziehbar.

3.3 Ein kooperatives Modell der Funktionsaufteilung nach dem Vorbild von Konfigurationsproblemen

Die in Abschnitt 3.2 festgestellten Interaktionsdefizite sind auf eine unzureichende Umsetzung des Prinzips der interaktiven Optimierung bzw. der kooperativen Funktionsaufteilung zurückzuführen (vgl. Abschnitt 1.3.2). Es fehlt eine Computerunterstützung für die manuelle Modifikation von Lösungseigenschaften. Im Folgenden werden Unterstützungsmöglichkeiten untersucht, die aus dem Anwendungsbereich der Konfigurationsprobleme stammen.

3.3.1 Gemeinsamkeiten zwischen semi-strukturierten Planungsproblemen und Konfigurationsproblemen

Bei Konfigurationsproblemen steht, ähnlich wie bei semi-strukturierten Planungsproblemen, die Interaktion mit dem Anwender im Vordergrund. Als Konfiguration wird dabei ein Vorgang bezeichnet, bei dem ein Kunde mitbestimmen kann, welche Merkmale ein gewünschtes Produkt bzw. seine Teilkomponenten besitzen sollen. Die Kombinationsmöglichkeiten der Merkmale sind durch bestimmte Vorgaben beschränkt. Produkte können z.B. Autos, Möbel oder Rechner sein (Stormer, 2007). Konfiguationsprobleme können als CSPs aufgefasst werden. Bei einem CSP mit $CSP = (V, D, C)$ repräsentieren die Variablen V die Attribute eines Produkts bzw. die Teilkomponenten eines zusammengesetzten Produkts (z.B. Tischbein, Tischplatte). Die zugehörigen Domänen enthalten die möglichen Attributwerte bzw. die möglichen Ausprägungen der Teilkomponenten (z.B. „rund“ oder „eckig“ für das Attribut Tischbein). Die Randbedingungen C legen Bedingungen fest, die von den Attributen erfüllt werden müssen.

Beispiel 3.2 (Fehn, 2014): Das in Beispiel 1.9 beschriebene Konfigurationsprobleme kann als CSP formuliert werden mit $V = \{\text{farbe, groesse, aufdruck}\}$ und

- $D_1 = \{\text{rot, schwarz, weiß}\}$,
- $D_2 = \{\text{klein,mittel,groß}\}$,
- $D_3 = \{\text{„Men in Black“, „Rettet die Wale!“}\}$

Es gelten die Randbedingungen $C = \{C_1, C_2\}$ mit

- $C_1 = (\text{aufdruck} = \text{“Men in Black“}) \Rightarrow (\text{farbe} = \text{schwarz})$ und
- $C_2 = (\text{aufdruck} = \text{“Rettet die Wale“}) \Rightarrow (\text{groesse} \neq \text{klein})$.

Gemeinsamkeiten zwischen Konfigurationsproblemen und kombinatorischen Planungsproblemen wurden bereits in Abschnitt 1.3.4 identifiziert. Wie bei Konfigurationsproblemen steht bei semi-strukturierten Planungsproblemen die Auswahl einer Lösung mit einer bestimmten Kombination von Eigenschaften im Vordergrund. Diese werden in beiden Fällen durch die Auswahl von einzelnen Werten

oder Teilmengen von Werten aus den endlichen Wertebereichen der Variablen vorgegeben (vgl. Gleichung 2.20). Die Variablen- und Wertauswahl erfolgt in einer nutzerspezifischen Reihenfolge. Der Nutzer kann in beiden Fällen in eine Sackgasse geraten, wenn sich die Zuweisung nicht mehr fortsetzen lässt und die partielle Spezifikation von Eigenschaften kein Bestandteil des Lösungsraums ist.

3.3.2 Die Richtlinien von Frayman (2001)

Um den Konfigurationsprozess für den Kunden so einfach wie möglich zu gestalten, wurden Richtlinien für die Funktionsaufteilung zwischen Mensch und Computer entwickelt. Im Folgenden werden 7 Richtlinien vorgestellt, die zurückgehend auf Frayman (2001) von Madsen (2003) (Seite 39) formuliert wurden (es folgt eine Übersetzung):

1. *„Wenn die Entscheidungsunterstützungsaufgabe abgeschlossen ist, muss die resultierende Instanziierung konsistent sein. Diese Anforderung ist die Motivation für die Implementierung eines Constraint-basierten Entscheidungsunterstützungssystems.*

2. *Der Anwender sollte ausdrücken können, dass „diese Variable nicht diese Werte besitzen sollte“. Oft ist nicht ein bestimmter Wert einer Variablen interessant, sondern es ist wichtig, dass ihr bestimmte Werte nicht zugewiesen werden.*

3. *Wenn der Anwender eine Auswahl treffen möchte, die mit den vorherigen Zuweisungen inkonsistent ist, sollte das System dem Anwender eine Liste mit den vorherigen Zuweisungen anbieten, die zurückgenommen werden müssen, um die neue Auswahl konsistent zu machen. Auf diese Weise wird der Anwender nicht durch die vorherigen Zuweisungen eingeschränkt, da sie geändert werden können, wenn ein bestimmter Wert einer anderen Variable als wichtiger erachtet wird.*

4. *Der Anwender sollte in der Lage sein, eine Auswahl zu treffen und später wieder zurückzunehmen. Es ist ein fundamentaler Aspekt einer guten Benutzeroberfläche dass der Anwender Aktionen rückgängig machen kann, denn dies erlaubt es, ohne Risiko mit Aktionen zu experimentieren.*

5. *Der Anwender muss eine Auswahl in beliebiger Reihenfolge treffen können. Auch dies ist ein fundamentaler Aspekt einer guten Benutzerschnittstelle, denn unterschiedliche Anwender können eine unterschiedliche Reihenfolge von Aktionen bevorzugen.*

6. *Der Anwender sollte keine Auswahl treffen können, die in eine Sackgasse führt, d.h. in eine Situation, in der eine Lösung aufgrund der bisherigen Zuweisungen nicht gefunden werden kann. [...]*

7. *Die Antwortzeit sollte für alle Funktionen kurz sein. Eine Funktion sollte weniger als 1 Sekunde benötigen, um den Gedankenfluss des Anwenders nicht zu unterbrechen. Um die Aufmerksamkeit des Anwenders zu beizubehalten, sollte eine Wartezeit von 10 Sekunden nicht überschritten werden (Nielsen, 1994).“*

Ein Teil der Richtlinien spiegelt die Grundannahmen des Lösungsschemas in Algorithmus 5 und 6 wieder: Richtlinie 1 umfasst das übergeordnete Ziel jedes Planungsvorgangs, nämlich die in Bezug auf das Planungsmodell gültige Umsetzung unstrukturierter Anforderungen. Die Richtlinien 4 und 5 werden im Lösungsschema durch eine freie Variablenauswahl repräsentiert, bei der auch die wiederholte Betrachtung einer Variablen nicht ausgeschlossen ist. Eine kurze Antwortzeit des Computers nach Richtlinie 7 wurde in Abschnitt 3.2.1 als Anforderung definiert.
Die Richtlinien 2, 3 und 6 formulieren zusätzliche Anforderungen, mit denen sichergestellt werden soll, dass der Lösungsprozess erfolgreich und mit geringem Aufwand für den Anwender vonstatten geht. So wird in Richtlinie 6 die Anforderung gestellt, dass Sackgassen nach Möglichkeit beseitigt werden sollten. Dazu muss der Computer inkonsistente Werte aus den Domänen der Variablen entfernen.
Um die Freiheit des Anwenders nicht einzuschränken, muss ihm jedoch auch die Auswahl inkonsistenter Werte gestattet werden. Die Suche nach Zuweisungen, die zur Wiederherstellung der Konsistenz zurückgenommen bzw. geändert werden müssen, sollte dabei nach Richtlinie 3 vom Computer unterstützt werden. Der Arbeitsaufwand für den Anwender wird dadurch reduziert, denn es wird verhindert, dass er Zuweisungen betrachtet, die nicht die Ursache für den aktuellen Konflikt sind (vgl. Abschnitt 3.2.1).
Als Grundlage für die Diskussion im nächsten Abschnitt werden nun Umsetzungsmöglichkeiten für die Richtlinien 3 und 6 untersucht.

Ansatz für die Umsetzung der Richtlinie 3

Zur Umsetzung von Richtlinie 3 muss mindestens eine (in der Regel möglichst kleine) Menge von Wertzuweisungen bzw. Randbedingungen identifiziert werden, die geändert werden müssen, damit die vom Anwender gewünschte Zuweisung ermöglicht wird. Diese sogenannte „Konfliktmenge“ repräsentiert eine Erklärung für den Konflikt, d.h. für die Inkonsistenz der gewünschten Wertzuweisung. Jussien und Barichard (2000) beschreiben, wie die Informationen, die dem im System verwendeten Constraintlöser zur Verfügung stehen, genutzt werden können, um dem Anwender eine oder mehrere Erklärungen für einen Konflikt bereitzustellen.

Ansätze für die Umsetzung der Richtlinie 6

Eine naive Umsetzung der Richtlinie 6 könnte wie folgt aussehen: Der Computer weist der Variable v_{select} probeweise alle Werte aus D_{select} zu. Die partielle Lösung wird jeweils um die Wertzuweisung für V_{select} erweitert und es erfolgt intern ein Aufruf von $computerSolve(assigned, unassigned)$. Dabei wird versucht, eine vollständige Lösung zu erzeugen. Wenn eine Lösung gefunden wird, kann der Wert im Wertebereich verbleiben. Existiert keine Lösung, so wird der Wert als inkonsistent identifiziert und entfernt.
Es wird deutlich, dass dieses Vorgehen im schlechtesten Fall einen exponentiellen Rechenaufwand erfordert, da für viele Planungsprobleme nicht in polynomieller Zeit entschieden werden kann, ob eine Lösung existiert oder nicht. Eine Garantie für eine kurze Antwortzeit des Computers kann damit nicht gegeben werden (Richtlinie 7). Um den Rechenaufwand und damit die Antwortzeit während der interaktiven Lösung zu reduzieren, wurden Techniken zur Verarbeitung des Constraint-Netzes entwickelt. Dabei spielen folgende Strategien eine wesentliche Rolle:

1. Eine Vorverarbeitung des Constraint-Netzwerks, sodass das Problem anschließend in polynomieller Zeit ohne Backtracking gelöst werden kann (eine sogenannte „Offline“-Verarbeitung).

2. Die Propagierung der Wertzuweisungen des Anwenders, d.h. die Filterung der Domänen oder die Wiederherstellung von k-Konsistenz für ein bestimmtes k nach jedem Zuweisungsschritt (vgl. Abschnitt 2.3.6).

Strategie 1 erfordert in der Regel eine strukturelle Zerlegung des CSPs, d.h. die Erzeugung eines Constraint-Graphen, dessen spezielle Struktur eine Problemlösung in polynomieller Zeit ermöglicht. Es wird eine azyklische Struktur angestrebt, die für ein beliebiges CSP z.B. mit der Methode des Baum-Clustering erzeugt werden kann (Gottlob, Leone und Scarcello, 2000; Dechter und Pearl, 1989). Eine weitere Methode ist die Herstellung eines azyklischen Graphen in Form eines binären Entscheidungsdiagramms (Hadzic und Andersen, 2004). Für ein auf diese Weise vorverarbeitetes CSP erfordert die Computerunterstützung der Richtlinie 6 (und ebenso der Richtlinie 3) lediglich einen polynomiellen Aufwand, sodass die Einhaltung der maximalen Antwortzeiten nach Richtlinie 7 möglich ist (Madsen, 2003). Nach der strukturellen Zerlegung des CSPs ist in der Regel die Einhaltung einer festen Variablenreihenfolge erforderlich. Bei Hadzic und Andersen (2004) wird jedoch (am Beispiel von Konfigurationsproblemen) gezeigt, wie binäre Entscheidungsdiagramme eingesetzt werden können, um dem Anwender eine effiziente, backtrackingfreie Interaktion zu ermöglichen, bei der die Reihenfolge der Variablenauswahl beliebig ist. Ähnliche Strategien zur Herstellung von Backtracking-Freiheit werden bei Amilhastre, Fargier und Marquis (2002), Hadzic u. a. (2004) und Fargier und Vilarem (2004) beschrieben.
Nach Strategie 2 müssen inkonsistente Werte aus den Domänen der Variablen entfernt werden. Dazu werden schnelle Verfahren zur Constraint-Propagierung und Filterung benötigt, wobei in der Regel nicht alle inkonsistenten Werte identifiziert werden können (Abschnitt 2.3.6). Das Auftreten von Sackgassen kann somit reduziert, aber nicht vollständig verhindert werden (lokale Konsistenz).

3.3.3 Diskussion: Eignung der Richtlinien für Planungsprobleme

Die in Abschnitt 3.3.3 vorgestellten Richtlinien stellen auch für die Bearbeitung semi-strukturierter Planungsprobleme eine vorteilhafte Funktionsaufteilung dar. Mit ihnen können herkömmliche Interaktionskonzepte im Sinne einer interaktiven Optimierung (vgl. Abschnitt 1.3.2) erweitert werden. Während die Richtlinien 1, 4, 5 und 7 unverändert übernommen werden können, müssen die Richtlinien 2, 3 und 6 an die besonderen Anforderungen von Planungsproblemen angepasst werden. Welche Anpassungen notwendig sind, wird im Folgenden erläutert.

Anpassung von Richtlinie 2

Die Feststellung, dass nicht immer „ein bestimmter Wert einer Variablen interessant“ ist, gilt auch für Planungsprobleme. Hier liegen jedoch häufig sehr große Wertebereiche[5] vor. Das Expertenwissen des Disponenten führt oft dazu, dass nur ein bestimmter Teilbereich des Wertebereiches bevorzugt wird. Das Planungssystem muss daher eine effiziente Auswahl von Teilbereichen ermöglichen. Wenn der Disponent stets den größtmöglichen tolerierbaren Teilbereich wählt, werden die Gefahr der Verschlechterung der Qualität und der Planung „nach Versuch und Irrtum“ reduziert (vgl. Abschnitt 3.2.3), da der Computer mehr Handlungsspielraum bei der Suche nach einer guten Lösung erhält.

Anpassung von Richtlinie 3

Nach Richtlinie 3 ermittelt der Computer eine Liste von Zuweisungen, die vom Anwender zurückgenommen bzw. geändert werden müssen. Für sehr große Planungsprobleme mit mehreren 100 oder 1000 Aufgaben ist diese Strategie nicht geeignet. Eine solche Liste würde u.U. viele Zuweisungen enthalten, die für den Disponenten bei der Umsetzung bestimmter unstrukturierter Anforderungen nicht relevant sind. Es kann sich z.B. um Zuweisungen handeln, die vom Computer automatisch ermittelt worden sind. Demzufolge würde eine Bearbeitung dieser Zuweisungen den Arbeitsaufwand des Disponenten unnötig erhöhen. Stattdessen ist es sinnvoll, den Disponenten über das geschätzte Ausmaß der Umplanung in den einzelnen Regionen des Plans zu informieren. Dabei sollte von konkreten Zuweisungen abstrahiert werden. Auf diese Weise wird es dem Disponenten ermöglicht, die Konsequenzen einer gerade betrachteten (inkonsistenten) Zuweisung abzuschätzen.
Des Weiteren ist es problematisch, dass der Computer nach Richtlinie 3 lediglich ermittelt, *welche* Zuweisungen geändert werden müssen. Die Entscheidung darüber, *wie* die Zuweisungen geändert werden müssen, bleibt jedoch dem Anwender überlassen. Der Handlungsspielraum zur Korrektur der Zuweisungen ist jedoch besonders bei Planungsproblemen u.U. sehr groß und bei der Änderung der Zuweisungen können u.U. erneut Sackgassen auftreten. Eine manuelle Umplanung ist also möglicherweise sehr aufwändig und erfordert, dass sich der Disponent mit für

[5] Beispiel: die Startzeit einer Aufgabe kann zwischen 8 und 16 Uhr liegen, wobei der Planungshorizont in Minuten unterteilt ist.

ihn nicht relevanten Zuweisungen auseinandersetzen muss. Zudem besteht die Gefahr, die Qualität des Plans zu verschlechtern (Abschnitt 3.2.3). Um den Arbeitsaufwand zu reduzieren, sollte der Computer den Disponenten bei der Umplanung unterstützen.
In der Literatur werden dazu verschiedene Implementierungsansätze beschrieben, die auf Konfigurationsprobleme angewendet wurden (vgl. Amilhastre, Fargier und Marquis (2002); Freuder und O'Sullivan (2001); O'Callaghan, O'Sullivan und Freuder (2005); Pu und Faltings (2004)). Sie basieren im Wesentlichen darauf, die vorherigen Zuweisungen des Anwenders als weiche Randbedingungen aufzufassen. Es erfolgt eine Optimierung nach einer Zielfunktion, mit der die Beibehaltung möglichst vieler dieser Zuweisungen angestrebt wird.
Bei Planungsproblemen sollte die Umplanung in ähnlicher Weise unterstützt werden. Im Unterschied zu Konfigurationsproblemen ist es bei Planungsproblemen jedoch u.U. nicht wünschenswert, dass Variablen, deren Werte nicht beibehalten werden können, vom Computer einen beliebigen neuen Wert (aus der Domäne) erhalten (vgl. Abschnitt 3.2.3, „Mangelnde Kontrolle über das Ausmaß der Umplanung"). Stattdessen besitzt der Disponent in der Regel Präferenzen darüber, wie stark die neuen Werte von den alten Werten abweichen dürfen. Er möchte z.B. die Umplanung von Aufgaben auf bestimmte Zeitabschnitte begrenzen. Wenn zahlreiche Aufgaben von der Umplanung betroffen sind, ist es für den Disponenten jedoch sehr aufwändig, für jede Aufgabe den entsprechenden Wertebereich manuell zu begrenzen (vgl. Richtlinie 2). Das Planungssystem sollte dem Disponenten daher effizientere Interaktionsmöglichkeiten bieten, um das Ausmaß einer automatischen Umplanung zu kontrollieren.

Anpassung von Richtlinie 6

Die Eliminieren sämtlicher Sackgassen durch Vorberechnung des Lösungsraums bzw. durch die Herstellung globaler Konsistenz ist für praktische Planungsprobleme nicht möglich. Bei der Gestaltung der Funktionsaufteilung können daher Situationen, in denen sich der Disponent in einer Sackgasse befindet, nicht ausgeschlossen werden.
Dieser Kritikpunkt resultiert aus der Größe praktischer Planungsprobleme. Es handelt sich häufig um *große* Optimierungsprobleme mit mehreren 1000 Variablen. Eine Vorberechnung des Lösungsraums würde für ein NP-schweres Problem dabei

u.U. nicht nur mehrere Stunden, sondern mehrere Tage oder Wochen in Anspruch nehmen. Dieser Rechenaufwand wäre zum einen aus Kostengründen nicht vertretbar, zum anderen könnten kurzfristige taktische Entscheidungen, die zu einer Veränderung des Planungsmodells und damit des Lösungsraums führen würden, nicht in die Berechnung einbezogen werden. Die Praxistauglichkeit der Pläne wäre somit beeinträchtigt. Zudem steht ein derart großer Rechenaufwand in keinem angemessenen Verhältnis zu dem erwartungsgemäß geringen Anteil an Variablen, die bei der wissensbasierten Planung tatsächlich vom Disponenten betrachtet werden müssen.
Eine in der Regel mit vertretbarem Aufwand realisierbare Strategie ist dagegen die Propagierung der Planungsentscheidungen des Disponenten nach jedem Zuweisungsschritt. Es kann versucht werden, so viele inkonsistente Werte wie möglich aus den Domänen der noch nicht zugewiesenen Variablen zu entfernen, um die Wahrscheinlichkeit für das Auftreten von Sackgassen zu reduzieren. Die Anwendung dieser Strategie im interaktiven Planungsprozess wird in Abschnitt 3.4 erläutert.

3.4 Planungswerkzeuge für die allgemeine Modellstruktur

In diesem Abschnitt wird das kooperative Modell der Funktionsaufteilung für Planungsprobleme spezialisiert. Dabei steht die Computerunterstützung bei der interaktiven Anpassung partieller oder vollständiger Pläne im Vordergrund, insbesondere werden die Richtlinien 2, 3 und 6 nach Frayman an die Anforderungen von Planungsproblemen angepasst bzw. erweitert (Abschnitt 3.3.3). Aus der allgemeinen Modellstruktur für Planungsprobleme werden 5 Planungswerkzeuge hergeleitet, die im interaktiven Lösungsprozess genutzt werden können. Mit ihnen kann der Disponent einen Plan nach seinen Vorstellungen entwickeln und anpassen.

3.4.1 Vorüberlegungen

Bisher wurden im interaktiven Lösungsprozess alle Variablen „anonymisiert“ betrachtet. Um die Funktionsaufteilung speziell an die Bedürfnisse von Disponenten

anzupassen, muss nun die allgemeine Struktur von Planungsmodellen berücksichtigt werden. Sie enthält zwei wesentliche Variablentypen: Startzeitvariablen und Ressourcenvariablen. Sie bilden die beiden Dimensionen, entlang derer sich der Planungsvorgang abspielt: die zeitliche Dimension und die Dimension der Ressourcen. An ihnen orientiert sich in der Regel auch die grafische Darstellung von Plänen. Der Planungsvorgang besteht aus Sicht des Disponenten darin, jeder betrachteten Aufgabe $a \in AS$ eine Position auf der Zeit- und Ressourcenachse zuzuordnen (vgl. Abbildung 3.2).
Die Wahl einer Position auf der Zeitachse entspricht dabei einer Wertzuweisung aus dem Planungshorizont P für die Variable a_{start}. Im Folgenden wird davon ausgegangen, dass jede Aufgabe eine feste Ausführungsdauer $dur(a)$ besitzt und stets die Randbedingung 2.9 gilt. In diesem Fall ist jede Wahl einer Startzeit gleichzeitig mit der Festlegung des Endzeitpunktes verbunden.
Auf der Ressourcenachse wird die Wertzuweisung für die Variable a_{res} vorgenommen. Es können sowohl einzelne Ressourcen, als auch Kombinationen von Ressourcen, die gleichzeitig an der Ausführung der Aufgabe beteiligt sind, zur Auswahl gestellt werden. Im Folgenden werden zur Vereinfachung stets einzelne Ressourcen dargestellt, d.h. die Wertauswahl für a_{res} erfolgt direkt aus RS. Die Interaktionskonzepte sind auf Ressourcenkombinationen übertragbar.

Unterteilung eines Plans in Abschnitte

Häufig wird die Zeit- und die Ressourcenachse zur Orientierung für den Disponenten in einzelne Abschnitte unterteilt (vgl. Abbildung 3.2). In der zeitlichen Dimension kann z.B. eine Unterteilung in Tage oder Tageszeiten vorgenommen werden. In der Dimension der Ressourcen entspricht die Unterteilung in der Regel den vorhandenen Ressourcengruppen. Jeder Zeitabschnitt und jede Ressourcengruppe kann hierarchisch in weitere Unterabschnitte und Untergruppen unterteilt werden. Mögliche Abschnittsunterteilungen werden in Beispiel 3.3 und 3.4 vorgestellt.

[6]Erklärung zum Bild: In horizontaler Richtung werden die Startzeiten, in vertikaler Richtung die Ressourcen zugewiesen. Der Planungshorizont P ist in zwei Intervalle I1 und I2 mit jeweils 3 Unterintervallen UI1 bis UI3 bzw. UI4 bis UI6 unterteilt. Es existieren zwei Ressourcengruppen RG1 und RG2 mit je zwei Ressourcen R1 und R2 bzw. R3 und R4 (RGS sei die Menge der verfügbaren Ressourcengruppen). Graue Kästchen repräsentieren Aufgaben (hier mit konstanter Dauer). Zur Vereinfachung werden nicht mehr als zwei Aufgaben pro Zeitab-

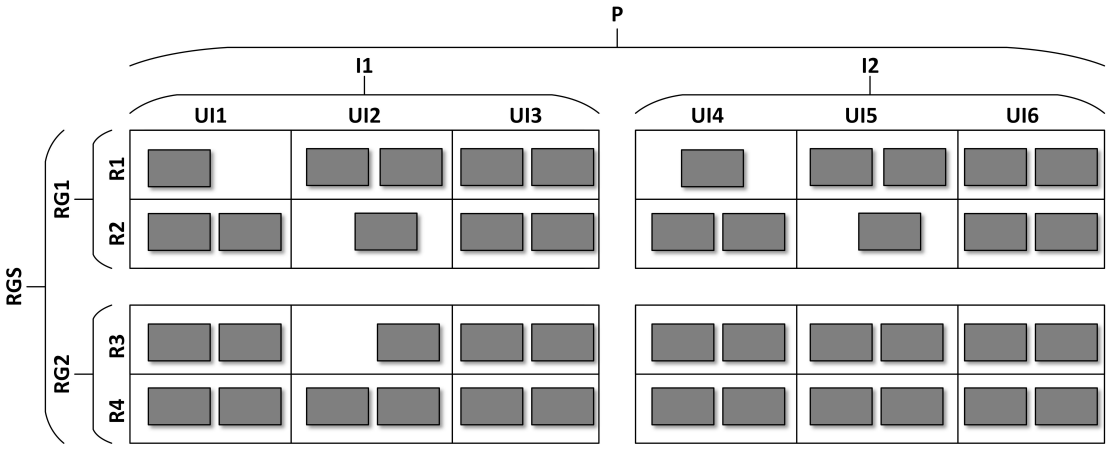


Abbildung 3.2: Schema für den Aufbau eines Plans[6].

Beispiel 3.3 (hierarchische Unterteilungen des Planungshorizontes):
Flottenplanung: Arbeitstage → Tageszeiten → Startzeiten (auf die Minute genau)
Es wird zwischen den Tageszeiten Vormittag (8-12 Uhr) und Nachmittag (12-20 Uhr) unterschieden.
Stundenplanung: Wochentage → Blöcke (z.B. 8-9:30 Uhr, 10-11:30 Uhr)
Jeder Wochentag wird in eine bestimmte Anzahl von Blöcken unterteilt, die jeweils 90 Minuten umfassen.
Schichtplanung: Arbeitstage → Schichten (z.B. Frühschicht)

Beispiel 3.4 (hierarchische Gruppierungen der Ressourcen):
Flottenplanung: Unterteilung der Fahrzeuge in Gruppen mit unterschiedlicher Beladungskapazität
Stundenplanung: Unterteilung der Räume nach Gebäuden → Ausstattungen (z.B. PC-Kabinett) → Anzahl der Sitzplätze
Produktionsplanung: Unterteilung der Maschinen nach Standorten → Abteilungen → Maschinentypen → Produktionsgeschwindigkeiten

schnitt dargestellt.

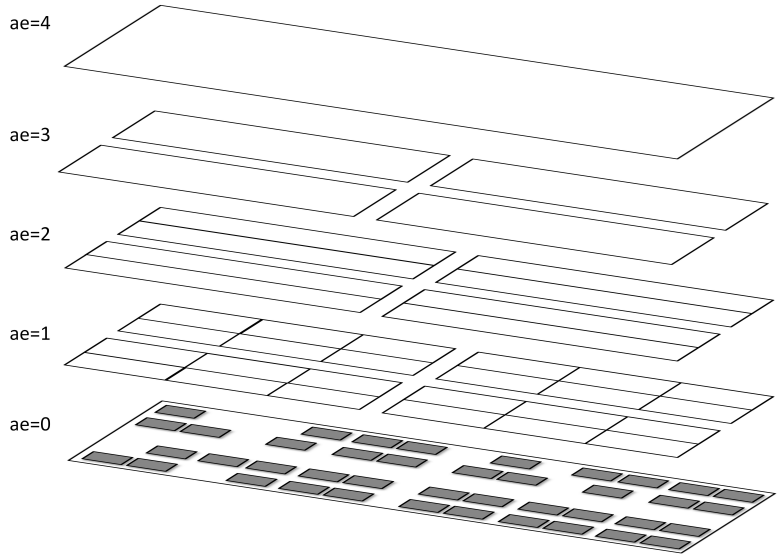


Abbildung 3.3: Verschiedene Abstraktionsebenen eines Plans

Unterteilung eines Plans in Abstraktionsebenen

Ein Abschnitt im Plan wird durch eine Kombination eines Abschnitts auf der Zeitachse mit einem Abschnitt auf der Ressourcenachse definiert. Der in Abbildung 3.2 gezeigte Plan enthält z.B. die Abschnitte (UI1,R1), (UI1,R2), (UI2,R1). Die Abschnitte ermöglichen es, einen zugrunde liegenden Plan mit einer geringeren Granularität abzubilden. Dabei werden die konkreten Positionen auf der Startzeit- und ggf. auch auf der Ressourcenachse vernachlässigt und nur die Abschnitte einer bestimmten Hierarchieebene betrachtet. Jeder Abschnitt repräsentiert aggregierte Informationen über den Teilplan, der von den darin befindlichen Aufgaben gebildet wird, ohne dass dieser Teilplan explizit an der Benutzerschnittstelle dargestellt werden muss. Ebenen mit niedrigerer Granularität bzw. höherer Abstraktion entstehen dadurch, dass jeweils eine Menge von Abschnitten auf der Zeit- und/oder Ressourcenachse zu einem übergeordneten Abschnitt zusammengefasst wird. Abbildung 3.3 zeigt beispielhaft 5 Abstraktionsebenen (ae) eines Plans mit unterschiedlicher Granularität. Die Anzahl der Ebenen und die Anordnung und Größe der Abschnitte (z.B. horizontal oder vertikal) kann für jedes Planungsproblem frei festgelegt werden (vgl. Abschnitt 3.6). Jedes Planungsproblem besitzt mindestens 2 Ebenen, auf denen die konkreten Positionen bzw. der Gesamtplan betrachtet wird.

Wenn für eine bestimmte Aufgabe a eine (neue) Position im Plan gesucht wird, ist eine Betrachtung des Plans in seiner höchsten Granularität, d.h. mit konkreten

Positionen ($start \in P, res \in RS$), u.U. nicht erforderlich. Bei einer niedrigeren Granularität sind die einzelnen Positionen gröber aufgelöst. Es kann davon ausgegangen werden, dass es in vielen Fällen ausreichend ist, Planungsentscheidungen in einem „grob aufgelösten“ Plan zu treffen. Dabei ist z.B. die Zuweisung eines Planabschnitts „(UI1,R1)“ ausreichend[7], während von der konkreten Startzeit abstrahiert werden kann. Dies kann bei der Suche nach einer vollständigen Lösung vorteilhaft sein, denn die Zuweisung eines Abschnitts grenzt den Wertebereich der betroffenen Variablen weniger ein als eine Zuweisung eines konkreten Wertes.
Im Folgenden wird bei der Konzeption der Werkzeuge I bis V vorausgesetzt, dass der Disponent einer Aufgabe a sowohl konkrete Positionen auf Ebene 0 als auch abstraktere Positionen auf Ebene 1 bis m zuweisen kann (m sei die Anzahl der Abstraktionsebenen).

3.4.2 Werkzeug I: Fixierung

Im interaktiven Lösungsschema nach Algorithmus 5 wird davon ausgegangen, dass das Planungssystem die Zuweisungen des Disponenten als Randbedingungen berücksichtigt. Die vorliegende partielle Lösung (Liste *assigned*) wird stets vom Computer unverändert beibehalten. Zuweisungen, die automatisch ermittelt werden sollen, müssen vom Disponenten gelöscht werden (Algorithmus 6, Zeilen 4-5). Im Folgenden werden Planungswerkzeuge konzipiert, mit denen auch partielle oder vollständige Lösungen automatisch angepasst werden können. Sie dienen z.B. der Auflösung von Konflikten, die durch die Verletzung von Randbedingungen entstanden sind. Da auch bei einer automatischen Anpassung die Vorstellungen des Disponenten berücksichtigt werden müssen, wird zunächst ein Planungswerkzeug benötigt, mit dem einzelne Zuweisungen als unveränderbar gekennzeichnet und somit von der Anpassung ausgeschlossen werden können. Diese Art der Kennzeichnung wird im folgenden als *Fixierung* bezeichnet.
Eine Fixierung umfasst die Auswahl eines bestimmten Wertes oder Teilbereiches von Werten aus der Domäne einer Variable und dessen Kennzeichnung als harte Randbedingung. *Unfixierte Zuweisungen, die Teil einer (partiellen) Lösung sind, gelten dagegen im Folgenden nicht mehr als harte Randbedingungen und dürfen daher bei der automatischen Planung (Algorithmus 5, Zeilen 6 und 15) bzw. von*

[7] Sowohl der Wert von a_{start}, als auch der Wert von a_{end} muss sich innerhalb des Abschnitts befinden.

den entsprechenden Planungswerkzeugen modifiziert werden. Alle nachfolgend beschriebenen Planungswerkzeuge berücksichtigen die Fixierung.
Die von einem Planungssystem gebotenen Fixierungsmöglichkeiten für die Variablen a_{start} und a_{res} einer Aufgabe $a \in AS$ sollten sich auf den einzelnen Abstraktionsebenen eines Plans an der Abschnittsunterteilung des Planungshorizontes und der Ressourcen orientieren. Auf Ebene 0 sollte die Fixierung konkreter Werte ermöglicht werden. Auf höheren Ebenen sollte die Fixierung von Abschnitten auf der Zeit- oder Ressourcenskala möglich sein. Bei der Fixierung eines Abschnitts auf der Zeitachse, der den Zeitraum $\{s_{start}, .., s_{end}\}$ umfasst, gilt für die Variable a_{start} die Einschränkung $a_{start} \in \{s_{start}, .., s_{end} - dur(a)\}$. Die Fixierung von konkreten Positionen oder Abschnitten, die von Werkzeug II bzw. Werkzeug IV (vgl. Kategorie 4) als inkonsistent bezüglich der ursprünglichen Randbedingungen identifiziert wurden, sollte im Planungssystem nicht möglich sein.
Für jede Aufgabe sollten Startzeiten und Ressourcen einzeln oder gleichzeitig fixiert werden können.

3.4.3 Werkzeug II: Unterstützung bei der Wertauswahl (Ebene 0)

Der Computer sollte den Disponenten auf Abstraktionsebene 0 bei der Wertauswahl für eine Variable unterstützen. Im zweidimensionalen Planschema (Abbildung 3.2) entspricht eine Wertauswahl der Suche einer Position auf der Startzeit- und Ressourcenachse, d.h. einer Kombination ($start \in P, res \in RS$) als Zuweisung für das Variablenpaar (a_{start}, a_{res}) einer Aufgabe a. In Anlehnung an Richtlinie 6 aus Abschnitt 3.3.2 sollte der Disponent vor der Auswahl inkonsistenter Positionen gewarnt werden. Es wird davon ausgegangen, dass sowohl vor dem interaktiven Planungsprozess als auch nach jedem einzelnen Zuweisungsschritt für das Constraint-Netzwerk des Planungsproblems lokale Konsistenz hergestellt wird (Propagierung). Bei Aufgaben, die auf einer Abstraktionsebene $ae, ae > 0$ einem bestimmten Abschnitt zugewiesen worden sind, wird der durch den Abschnitt begrenzte Wertebereich der Startzeit- und Ressourcenvariablen bei der Konsistenzherstellung zugrunde gelegt. Unter der Voraussetzung, dass der aktuelle Zustand des Plans lokal konsistent ist, sollte Werkzeug II die Positionen des Plans wie folgt klassifizieren, wenn für eine bestimmte Aufgabe a auf Abstraktionsebene 0 eine (neue) Position gesucht wird:

- Die Position ist *inkonsistent bezüglich der ursprünglichen Randbedingungen* des Planungsproblems, d.h. ohne Berücksichtigung der bisherigen automatisch oder manuell vorgenommenen Wertzuweisungen und Fixierungen.

- Die Position ist zulässig gemäß der ursprünglichen Randbedingungen, aber *inkonsistent bezüglich der bisherigen Zuweisungen*, die im Lösungsprozess erfolgt sind. Im Lösungsraum des Planungsproblems existiert kein vollständiger Plan, der den bisher erstellten partiellen Plan fortsetzt und in dem a diese Position einnimmt. Die Zuweisung führt entweder unmittelbar zu einer Verletzung von Randbedingungen mit anderen Aufgaben im aktuellen (partiellen) Plan oder nach ein oder mehreren weiteren Planungsschritten in eine Sackgasse.

- Die Position $(start, res)$ ist *lokal konsistent bezüglich der bisherigen Zuweisungen*[8]. Es existiert möglicherweise ein vollständiger Plan, der den bisher erstellten partiellen Plan fortsetzt und in dem a die Position mit $a_{start} = start$ und $a_{res} = res$ einnimmt. Da es sich nicht um globale Konsistenz handelt, kann keine Garantie für die Existenz einer vollständigen Lösung gegeben werden. Es ist daher nicht ausgeschlossen, dass später aufgrund der Auswahl dieser Position Sackgassen auftreten.

Abbildung 3.4 zeigt einen Plan, in dem konsistente Positionen für eine neu einzufügende Aufgabe a (gekennzeichnet durch Mauszeiger) grün hervorgehoben sind (alle anderen, rot gekennzeichneten Positionen sind inkonsistent). Es wird angenommen, dass pro Abschnitt maximal zwei Aufgaben überlappungsfrei eingeplant werden dürfen. Insgesamt wurden in den Abschnitten UI1 und UI2 drei Zeitintervalle mit möglichen Startzeiten gekennzeichnet. Was passiert, wenn der Beginn der Aufgabe auf einen Zeitpunkt außerhalb dieser Intervalle gesetzt wird? In den meisten Fällen (z.B. in den Abschnitten UI1 und UI2) wird die Verletzung der Überlappungsfreiheit sofort deutlich. Abschnitt (UI3, R3) enthält jedoch scheinbar eine freie Position. Wenn die Aufgabe dorthin verschoben wird, verletzt der partielle Plan zunächst keine Randbedingungen. Die Position wurde in diesem Beispiel aber als inkonsistent gekennzeichnet, da eine der noch nicht verplanten Aufgaben, deren Startzeitdomäne auf UI3 begrenzt ist, (im Bild unterhalb des Plans dargestellt) genau diesen Zeitraum benötigt. Der Disponent wird diesen Konflikt erst bemerken,

[8] Im Folgenden wird der Begriff „konsistent“ gleichgesetzt mit dem Begriff „lokal konsistent“ (vgl. Diskussion in Abschnitt 3.3.2 - Richtlinie 6).

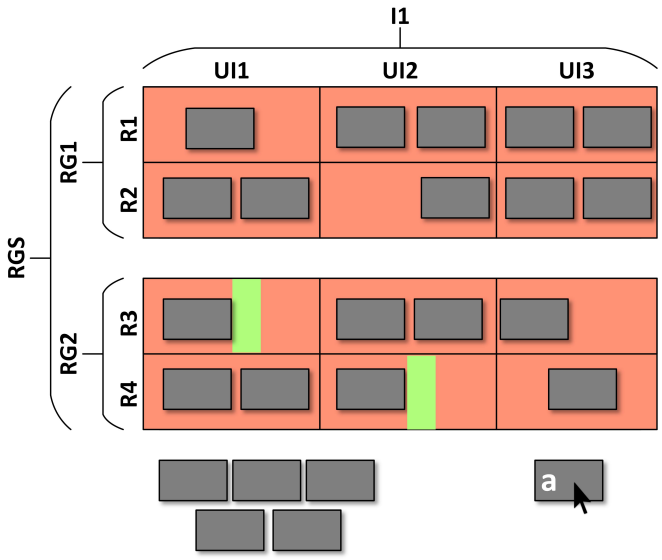


Abbildung 3.4: Wertauswahlunterstützung für eine bestimmte Aufgabe a durch Hervorhebung von konsistenten Positionen für diese Aufgabe im Plan.

wenn er diese Aufgabe nach beliebig vielen weiteren Zuweisungsschritten manuell oder automatisch einplanen möchte. Eine Umplanung anderer Aufgaben aus Abschnitt UI3 ist u.U. schwierig, wenn viele Positionen im Plan bereits besetzt sind. Es wird deutlich, dass eine vorausschauende Unterstützung bei der Wertauswahl benötigt wird, um den Disponenten vor Sackgassen bei der Planung zu bewahren. Aspekte der Implementierung von Werkzeug II werden in Abschnitt 3.5 diskutiert.

3.4.4 Werkzeug III: Planungsfunktionen

Werkzeug II ist für den wissensbasierten Planungsprozess unzureichend, da es den Umplanungsprozess zur Zuweisung von Positionen im Plan, die inkonsistent bezüglich der bisherigen Zuweisungen, aber trotzdem wünschenswert sind, nicht unterstützt. Im Folgenden wird ein Werkzeug konzipiert, welches eingesetzt werden kann, um manuelle Backtracking-Prozesse im interaktiven Lösungsprozess durchzuführen (vgl. „Anpassung von Richtlinie 3“ in Abschnitt 3.3.3). Die Unterteilung eines Plans in Abschnitte soll dafür die Grundlage bilden.

Definition des Werkzeugs

Wenn Planungsentscheidungen auf einem höheren Abstraktionsniveau getroffen werden, kann die Ermittlung konkreter Zuweisungen auf Ebene 0 automatisiert vom Computer vorgenommen werden. Zu diesem Zweck wird Werkzeug III konzipiert, welches auf jeder Abstraktionsebene $ae, ae > 0$ auf jeden Abschnitt angewendet werden kann: Es berechnet innerhalb eines gewählten Abschnitts einen Teilplan auf Ebene 0, der alle Aufgaben enthält, die innerhalb des Abschnitts auf den Ebenen $e, 0 \leq e \leq ae$ eingeordnet sind. Die Positionen der restlichen Aufgaben, die außerhalb des Abschnitts bereits auf Ebene 0 eingeplant sind, bleiben unverändert. Aufgaben, die noch unverplant sind bzw. sich nicht innerhalb des Abschnitts befinden, werden nicht berücksichtigt. Der berechnete Teilplan muss folgende Anforderungen erfüllen:

1. Die Zuweisungen innerhalb des Teilplans sind konsistent bezüglich der bisherigen Zuweisungen (gemäß der Klassifikation bei Werkzeug II). Es werden keine Randbedingungen verletzt, die zwischen den Aufgaben innerhalb des Abschnitts bestehen. Außerdem muss der Teilplan korrekt in den Gesamtplan eingebettet sein, d.h. es dürfen keine Randbedingungen verletzt werden, die zwischen Aufgaben innerhalb des Abschnitts und Aufgaben außerhalb des Abschnitts bestehen. Fixierte Zuweisungen werden als Randbedingungen betrachtet.

2. Der Teilplan dieses Abschnitts wird nach einer Heuristik bzw. Zielfunktion optimiert, mit der eine gute Qualität des Gesamtplans angestrebt wird (z.B. minimale Kosten, minimale Änderungen).

Es sei ae die Nummer der Abstraktionsebene, auf der Werkzeug III für einen bestimmten Abschnitt angewendet wird. Aufgaben, die innerhalb des Abschnitts auf einer tieferen Abstraktionsebene 1 bis $ae - 1$ einem bestimmten Unterabschnitt zugeordnet wurden, müssen sich nach der Anwendung von Werkzeug III genau dann in diesem Unterabschnitt befinden, wenn die Zuordnung zuvor fixiert wurde (Werkzeug I). Ebenso bleiben nach der Anwendung von Werkzeug III die Positionen von Aufgaben erhalten, die auf Ebene 0 fixiert wurden. Unfixierte Zuweisungen dürfen dagegen von Werkzeug III verändert werden.
Werkzeug III wird im Folgenden auch als *Planungsfunktion* bezeichnet. Planungsfunktionen sollten für alle Abschnitte auf allen Abstraktionsebenen eines Plans

bereitgestellt werden.
Mit Hilfe von Werkzeug III kann der Disponent lokal begrenzte Umplanungen durchführen. Folgendes Szenario ist z.B. denkbar: Für eine Aufgabe a wurde eine gewünschte Position auf Ebene 0 von Werkzeug II als inkonsistent bezüglich der bisherigen Zuweisungen klassifiziert. Der Disponent hat nun die Möglichkeit, die Aufgabe an dieser Position zu fixieren. Für den Abschnitt, in dem sich die Aufgabe auf einer gewählten Abstraktionsebene befindet, kann er anschließend die zugehörige Planungsfunktion aktivieren. Diese übernimmt die Umplanung der restlichen im Abschnitt befindlichen Aufgaben nach den oben definierten Anforderungen. Wurde die Aufgabe vorher aus einem anderen Abschnitt entfernt, muss ggf. auch dessen Planungsfunktion aktiviert werden, da der zugehörige Teilplan nun ggf. in Bezug auf die Qualität des Gesamtplans verbessert werden kann. Auch andere Szenarien sind denkbar. So kann der Disponent im gleichen Szenario auf die Fixierung von Aufgabe a verzichten, wenn er die Abschnittszuordnung, aber nicht die konkrete Position selbst festlegen möchte.

Umgang mit Konflikten

Die automatische Einplanung einer Aufgabe a in einen Abschnitt ist u.U. nicht möglich, da die Planungsfunktion keinen Teilplan erzeugen kann, der die oben definierten Anforderungen erfüllt. Der Disponent sollte in diesem Fall eine Fehlermeldung erhalten und der Zustand vor der Aktivierung von Werkzeug III sollte wiederhergestellt werden. Die Einplanung scheitert im Allgemeinen, wenn kein Teilplan ermittelt werden kann, in dem Aufgabe a eine konsistente Position einnimmt. Es lassen sich z.B. folgende Konfliktfälle unterscheiden:

1. Es existieren Randbedingungen, die das gleichzeitige Auftreten von Aufgaben innerhalb eines Zeitraums und/oder auf einer bestimmten Ressource verhindern. Beispiele dafür sind die Randbedingungen „Nicht alle Kurse der gleichen Lehrveranstaltung sollten am gleichen Tag stattfinden.“, „Kurse, die im gleichen Block liegen, dürfen nicht im gleichen Raum stattfinden.“ (Stundenplanung), „Innerhalb einer Tour dürfen sich nur Transportaufträge der gleichen Kategorie befinden.“ (Flottenplanung), „In der Frühschicht werden nicht mehr als 5 Krankenpfleger benötigt.“ (Schichtplanung). Wenn a im Abschnitt liegt, müssen daher ein oder mehrere andere Aufgaben aus diesem Abschnitt entfernt werden.

2. Der Abschnitt ist so „überfüllt“, dass eine konfliktfreie Anordnung der Aufgaben nach Anforderung 1 nicht mehr möglich ist. Es ist z.B. möglich, dass sich eine bestimmte Anzahl von Aufgaben mit einer bestimmten Dauer nicht mehr überlappungsfrei zu einer Tour anordnen lässt, deren maximale Länge durch ein Zeitfenster begrenzt ist. In diesem Fall müssen einige andere Aufgaben aus dem Abschnitt entfernt und an anderer Stelle eingeplant werden.

3. Einige Aufgaben innerhalb des Abschnitts (ggf. auch a selbst) stehen mit Aufgaben außerhalb des Abschnitts über Randbedingungen (z.B. Reihenfolgebedingungen) in Verbindung. Es existiert kein konsistenter, Aufgabe a einschließender Teilplan, bei dem diese Randbedingungen eingehalten werden. Bevor die Planungsfunktion erfolgreich angewendet werden kann, müssen daher folgende Maßnahmen getroffen werden: Ein oder mehrere Aufgaben außerhalb des Abschnitts werden umgeplant und/oder ein oder mehrere Aufgaben werden aus dem Abschnitt entfernt und in andere Abschnitte eingeplant.

4. Die Wahl jeder Position im Abschnitt führt dazu, dass für mindestens eine der noch nicht zugewiesenen Variablen kein gültiger Wert mehr verfügbar ist. Alle Positionen im Abschnitt sind für a also inkonsistent bezüglich der ursprünglichen Randbedingungen oder inkonsistent bezüglich der bisherigen Zuweisungen (vgl. Werkzeug II). In letzterem Fall kann der Disponent ggf. mit Hilfe der vorhandenen Planungswerkzeuge eine Konfliktbehandlung vornehmen, sodass die Einordnung von a in den Abschnitt wieder möglich wird. Der Disponent kann z.B. Aufgaben außerhalb des Abschnitts umplanen oder Aufgaben aus dem Abschnitt entfernen und in andere Abschnitte einplanen.

Welcher der Konfliktfälle 1 bis 3 vorliegt, ist für den Disponenten nachvollziehbar, wenn das Planungssystem an der Benutzerschnittstelle die Verletzung von Randbedingungen für die betroffenen Aufgaben kenntlich macht (vgl. Abschnitt 3.1.1). Der Disponent kann die Konfliktbehandlung vornehmen, indem er den betroffenen Aufgaben (z.B. mit Hilfe der zugehörigen Planungsfunktionen) neue Abschnitte zuweist. Bei Bedarf kann er auch auf Planungsfunktionen höherer Abstraktionsebenen zugreifen. Außerdem kann das später beschriebene Werkzeug V zur Konfliktbehandlung eingesetzt werden.
Konfliktfall 4 ist an der Benutzerschnittstelle nicht ohne Weiteres nachvollziehbar, da er erst in späteren Zuweisungsschritten zu einer Verletzung von Randbedingun-

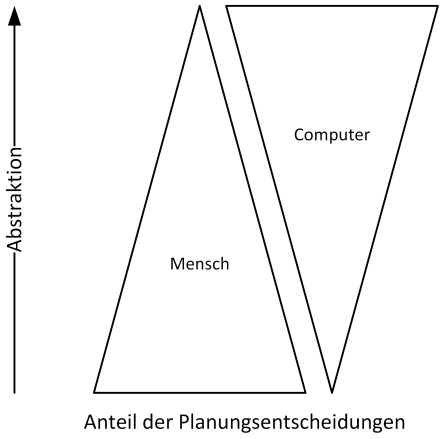


Abbildung 3.5: Anteil der Beteiligung von Mensch und Computer an der Ermittlung von Planungsentscheidungen auf unterschiedlichen Abstraktionsebenen

gen führt. Eine Kennzeichnung der Variablen, die aufgrund der Abschnittszuweisung einen leeren Wertebereich besitzen, kann die Nachvollziehbarkeit erhöhen.

Fazit

Mit Werkzeug III können Umplanungsvorgänge an den Computer delegiert werden, aber sie bleiben lokal begrenzt. Die Kontrolle darüber, welche Bereiche eines Plans von der Umplanung betroffen sind, verbleibt dadurch beim Disponenten. Je höher der Abstraktionsgrad gewählt wird, desto größer ist die Beteiligung des Computers an der Umsetzung einer Planungsentscheidung, d.h. an einer Abschnittszuweisung (Abbildung 3.5). Eine Lösungsalternative lässt sich also in der Entwurfsphase eines Entscheidungsprozesses mit dem geringstmöglichen Aufwand erzeugen, wenn vom Disponenten die höchstmögliche Abstraktionsebene gewählt wird, die zur Spezifikation einer unstrukturierten Anforderung ausreichend ist.

3.4.5 Werkzeug IV: Erweiterte Unterstützung bei der Wertauswahl

Werkzeug II ermittelt inkonsistente Positionen im Plan, deren Zuweisung eine Umplanung erfordert. Es soll nun für höhere Abstraktionsebenen verallgemeinert werden. Dabei werden die Informationen, die mit Hilfe der Planungsfunktionen

über das Ausmaß der notwendigen Umplanung gewonnen werden können, in die Entscheidungsunterstützung bei der Wertauswahl einbezogen. Es wird vorausgesetzt, dass sich der Plan vor der Anwendung von Werkzeug IV in einem lokal konsistenten Zustand befindet.
Die Unterstützung bei der Wertauswahl bezieht sich stets auf eine bestimmte Aufgabe a, für die eine (neue) Position im Plan gesucht wird. Auf höheren Abstraktionsebenen gilt zunächst: Jeder Abschnitt, der mindestens eine konkrete Position ($start \in P, res \in RS$) enthält, die von Werkzeug II als konsistent identifiziert wurde, kann insgesamt als konsistent markiert werden, denn die konkreten Positionen sind auf höheren Abstraktionsebenen nicht relevant. Wenn in einem Abschnitt keine konsistente Position existiert, sollte der Disponent darüber informiert werden, ob der Abschnitt konsistent vorbehaltlich der Anwendung der Planungsfunktion ist. Dies ist der Fall, wenn eine Umplanung innerhalb des Abschnitts möglich ist, um eine konsistente Position zu schaffen. Insgesamt können folgende Kategorien zur Kennzeichnung von Abschnitten auf einer Abstraktionsebene $ae, ae > 0$ identifiziert werden:

Kategorie 1: Der Abschnitt enthält mindestens eine konsistente Position für a (Beispielfarbe Grün).

Kategorie 2: Der Abschnitt enthält keine konsistenten Positionen für a. Mit Hilfe der zugehörigen Planungsfunktion können jedoch die anderen, bereits eingeplanten Aufgaben innerhalb dieses Abschnitts so umgeplant werden, dass für a eine konsistente Position frei wird (Beispielfarbe Gelb).

Kategorie 3: Der Abschnitt enthält keine konsistenten Positionen für a. Die Anwendung der zugehörigen Planungsfunktion wird scheitern, da einer der Konfliktfälle 1 bis 4 vorliegt (vgl. voriger Abschnitt). Der Abschnitt kann jedoch in Kategorie 1 bzw. 2 überführt werden, wenn (ggf. unter Zuhilfenahme der Planungswerkzeuge) Aufgaben aus dem Abschnitt entfernt bzw. Aufgaben außerhalb des Abschnitts umgeplant werden (Beispielfarbe Grau, vgl. Konfliktfälle 1 bis 4 im vorigen Abschnitt).

Kategorie 4: Der Abschnitt ist für a inkonsistent, da alle darin enthaltenen Positionen von Werkzeug II als inkonsistent bezüglich der ursprünglichen Randbedingungen identifiziert wurden (Beispielfarbe Rot). Es ist nicht möglich, durch Umplanung von Aufgaben innerhalb oder außerhalb des Abschnitts

eine konsistente Position zu schaffen.

Zur erweiterten Unterstützung der Wertauswahl sollte Werkzeug IV die Abschnitte eines Plans gemäß dieser Kategorien kennzeichnen. Auf der untersten Abstraktionsebene 0 ist darüber hinaus auch eine Kategorisierung konkreter inkonsistenter Positionen möglich. Dabei kann z.B. die Abschnittsbegrenzung der nächst höheren Abstraktionsebene zugrunde gelegt werden. Für jede Position, die inkonsistent bezüglich der bisherigen Zuweisungen ist, wird dann ermittelt, ob nach einer Fixierung der Aufgabe a an dieser Position der Teilplan dieses Abschnitts so korrigiert werden kann, dass die Position konsistent ist (entspricht Kategorie 2) oder ob dies nur möglich ist, nachdem andere Aufgaben aus dem Abschnitt entfernt wurden (entspricht Kategorie 3).
Wenn der Disponent eine Aufgabe a in einen bestimmten Abschnitt auf Abstraktionsebene $ae, ae > 0$ einplanen möchte, kann er aufgrund der Kategorisierung des Abschnitts zwischen folgenden Handlungsmöglichkeiten wählen:

- Er wechselt zu Ebene 0 und weist der Aufgabe eine Position zu, die von Werkzeug II als konsistent gekennzeichnet wurde (nur möglich bei Kategorie 1).

- Er weist die Aufgabe dem Abschnitt zu und aktiviert die zugehörige Planungsfunktion, um einen optimierten Teilplan erzeugen zu lassen, der diese Aufgabe enthält (bestehende Zuweisungen werden dabei u.U. verändert, möglich bei Kategorie 1 und 2).

- Er weist die Aufgabe dem Abschnitt zu und setzt die Planung zunächst mit anderen Aufgaben fort (möglich bei Kategorie 1, 2 und 3). Wenn eine Veränderung der Zuweisung durch Planungsfunktionen höherer Abstraktionsebenen bzw. von Werkzeug V verhindert werden soll, kann der Disponent die Zuweisung fixieren (Werkzeug I).

Abbildung 3.6 zeigt eine beispielhafte Kennzeichnung nach den Kategorien 1 bis 4 auf 4 Abstraktionsebenen eines Plans. Auf der untersten Ebene ($ae = 0$) werden konkrete Startzeit- und Ressourcenzuweisungen betrachtet. Positionen auf Ebene 0, die von Werkzeug II als konsistent identifiziert wurden, sind grün gekennzeichnet. Die Abschnittsunterteilung von $ae = 1$ wird auf $ae = 0$ eingeblendet. Dementsprechend werden die inkonsistenten Positionen auf Ebene 0 von Werkzeug IV in die Kategorien 2 bis 4 eingestuft. Die Abschnitte der höheren Ebenen werden in

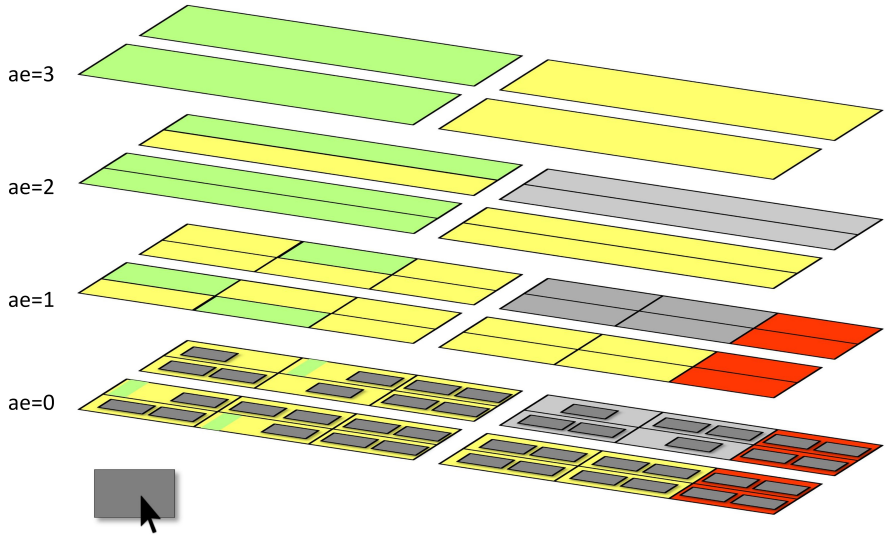


Abbildung 3.6: Verschiedene Abstraktionsebenen eines Plans mit beispielhafter Kennzeichnung der Abschnitte, wenn für eine bestimmte Aufgabe eine neue Position gesucht wird.

die Kategorien 1 bis 4 eingestuft.
Es wird deutlich, dass mehrere Abschnitte, die jeweils zu Kategorie 3 gehören (im Bild Beispielfarbe Grau auf $ae = 0$ bis $ae = 2$), auf einer höheren Abstraktionsebene ($ae = 3$) zu einem größeren Abschnitt der Kategorie 2 zusammengefasst werden können.

3.4.6 Werkzeug V: Unterstützung bei der Konfliktbehandlung

Wie Abbildung 3.5 zeigt, hängt die Beteiligung des Disponenten am Planungsprozess von der Wahl der Abstraktionsebene ab. Gleichzeitig wird mit sinkender Abstraktion ein größerer Anteil der Komplexität des Lösungs- und Optimierungsvorgangs auf die Seite des Disponenten verlagert.
Von Ebene $m - 1$ bis zur Ebene 0 (m sei die Anzahl der Ebenen) verringert sich die Größe der Planungsprobleme, die von den einzelnen Planungsfunktionen gelöst werden. Je niedriger die Ebene, desto größer ist für den Disponenten der Aufwand zur Behandlung der Konfliktfälle 1 bis 4 (vgl. Werkzeug III) und desto stärker treten die in Abschnitt 3.2.3 beschriebenen Defizite der manuellen Planung auf. Wenn für ihn die Planungsaufwand auf einer bestimmten Ebene zu hoch ist, kann er zur nächst höheren Ebene wechseln. Damit sinkt jedoch sein Einfluss auf den

Aufbau des Plans, da Planungsentscheidungen dann nur noch für gröber aufgelöste Positionen, d.h. in einer gröberen Abschnittsunterteilung, möglich sind[9]. Der Disponent muss einen Kompromiss zwischen dem Aufwand der Planung und dem Grad der Mitbestimmung finden.
Das fünfte Planungswerkzeug soll den Planungsaufwand für den Disponenten auf jeder Ebene reduzieren und für alle Ebenen auf einem nahezu konstanten Niveau halten. Es wird daher definiert als *abschnittsübergreifende Planungsfunktion, mit der die auf einer bestimmten Ebene auftretenden Konflikte so aufgelöst werden können, dass die Abschnittszuteilungen der Aufgaben weitestgehend erhalten bleiben.* Dem Disponenten wird es dadurch z.B. ermöglicht, ein oder mehreren Aufgaben Positionen zuzuweisen, die bezüglich der bisherigen Zuweisungen inkonsistent sind und nicht durch die Anwendung der Planungsfunktion für die jeweiligen Abschnitte konsistent gemacht werden können. Diese Positionen würden von Werkzeug IV nach Kategorie 3, Beispielfarbe Grau, gekennzeichnet. Werkzeug V sucht, bezogen auf eine gewählte Abstraktionsebene, einen Plan, der folgende Anforderungen erfüllt:

- Alle eingeplanten Aufgaben befinden sich an einer konsistenten Position.
- Alle eingeplanten Aufgaben, deren Positionen nicht mit Werkzeug I fixiert wurden, liegen im ursprünglichen Abschnitt der gewählten Ebene oder in einem Abschnitt dieser Ebene, der sich so nah wie möglich am ursprünglichen Abschnitt befindet. Wenn die unterste Abstraktionsebene gewählt wird, erfolgt eine Umplanung, sodass sich die Aufgaben möglichst nah an ihrer ursprünglichen Position befinden. Je höher die gewählte Abstraktionsebene, desto größer sind die Abschnitte und desto stärker können die Positionen folglich von den ursprünglichen Positionen abweichen.
- Alle eingeplanten Aufgaben, die vom Disponenten mit Werkzeug I auf einen bestimmten (ggf. inkonsistenten) Abschnitt fixiert wurden, befinden sich innerhalb des festgelegten Abschnitts (auf Ebene 1 bis $m-1$) bzw. an der festgelegten Position (auf Ebene 0). Je höher die Abstraktionsebene, auf der

[9]Die Möglichkeit der Fixierung wird hier zur Vereinfachung außer Acht gelassen. Eine Fixierung von Abschnitten auf unteren Abstraktionsebenen für bestimmte Aufgaben wird auch von Planungsfunktionen höherer Abstraktionsebenen berücksichtigt. Dadurch wird der Handlungsspielraum der automatischen Planung eingeschränkt, sodass Konflikte entstehen können, die vom Disponenten behandelt werden müssen. Fixierungen tragen also zur Erhöhung des Planungsaufwandes auf einer bestimmten Abstraktionsebene bei.

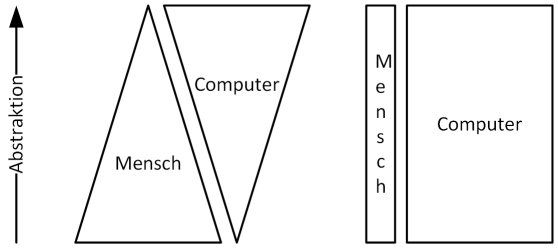


Abbildung 3.7: Verteilung des Planungsaufwandes zwischen Mensch und Computer auf unterschiedlichen Abstraktionsebenen. Links: ohne Werkzeug V, Rechts: mit Werkzeug V

die Fixierung erfolgt ist, desto größer ist der Abschnitt und desto größer ist folglich der Handlungsspielraum des Computers zur Positionierung dieser Aufgaben.

Der Aufwand einer manuellen Abschnittsumordnung entfällt damit für den Disponenten. Die Bereitstellung von Werkzeug V trägt somit dazu bei, dass für den Disponenten die Wahl der Abstraktionsebene nicht mehr vom Planungsaufwand, sondern nur vom benötigen Detaillierungsgrad zur Formulierung unstrukturierter Anforderungen abhängt (Abbildung 3.7). Wenn (z.B. aufgrund der gewählten Fixierungen) durch Werkzeug V keine Lösung gefunden werden kann, sollte der Disponent darüber benachrichtigt werden. In diesem Fall ist ggf. eine Anpassung der Fixierungen notwendig.

3.4.7 Anpassung der Werkzeuge für alternative Ansichten und Planungsstrategien

Die in diesem Abschnitt zugrunde gelegte Planansicht (zweidimensional, vertikal: Ressourcen, horizontal: Startzeiten) wird in industriellen Planungssystemen häufig genutzt (vgl. Abbildung 1.8). Es existieren jedoch alternative Darstellungsformen. Dazu gehören Ansichten, bei denen beide Achsen der Zuweisung von Startzeiten vorbehalten sind. Die Ressourcenzuweisung wird direkt auf den einzelnen Aufgaben gekennzeichnet (vgl. Abbildung 4.14).
Ob diese Darstellungsform für ein gegebenes Planungsproblem sinnvoll ist, kann wie folgt ermittelt werden: Zunächst sollte die für eine bestimmte Abstraktionsebene gewünschte Abschnittsaufteilung in das in Abbildung 3.2 gezeigte Plan-

schema eingezeichnet werden. Wenn die Abschnitte nur nach Zeitpunkten bzw. -Zeitintervallen unterteilt werden, aber nicht nach Ressourcen oder Ressourcengruppen (d.h. jeder Abschnitt umfasst alle Ressourcen), dann bietet es sich an, die Ressourcenachse in der grafischen Darstellung auszublenden. In diesem Fall erfolgt die Zuordnung von Aufgaben zu einem bestimmten Zeitpunkt stets unabhängig von der Ressourcenzuordnung (vgl. Fallstudie 2 in Kapitel 4).
Diese Darstellungsform unterstützt einen Planungsstil, bei dem der Disponent für eine Aufgabe zunächst nur die Startzeit festlegt und die Ressourcenzuordnung offen lässt. Die Ressourcenzuweisung kann vom Disponenten ggf. später, nach der Einplanung weiterer Aufgaben, nachgeholt werden (vgl. Hoc, Guerin und Mebarki (2012)). Auf diese Weise lässt sich bei der manuellen Planung eine zu frühe Einschränkung des Wertebereichs verhindern, die im weiteren Verlauf der Planung zu Sackgassen führen kann. Die einem Abschnitt zugehörige Planungsfunktion (Werkzeug III) kann eingesetzt werden, um die Ressourcenzuweisungen für alle Aufgaben eines Abschnitts automatisch zu vervollständigen.
Wenn dieser Planungsstil ermöglicht werden soll, ist eine Anpassung der Werkzeuge II und IV erforderlich. Werkzeug II ermittelt Vorschläge für Positionen $(start, res)$. Es kann abgewandelt werden, um dem Disponenten nur noch Vorschläge für Startzeiten anzuzeigen. Um bei der nachträglichen Ressourcenauswahl Konflikte zu vermeiden, sollten die Vorschläge für eine Aufgabe a wie folgt ermittelt werden (vgl. ursprüngliche Definition von Werkzeug II):

- Ein Position *start* wird als inkonsistent markiert, wenn sie inkonsistent ist.
- Eine Position *start* wird als inkonsistent markiert, wenn sie konsistent ist, aber alle Werte in der Domäne von a_{res} dazu führen, dass die Position $(start, res)$ inkonsistent ist.
- Ein Position *start* wird als konsistent markiert, wenn sie konsistent ist und wenn in der Domäne von a_{res} mindestens ein Wert res enthalten ist, sodass die Position $(start, res)$ konsistent ist.

Werkzeug IV sollte auf diese Kennzeichnungen konkreter Startzeiten zugreifen, um die Kennzeichnungen für die abstrakten Planabschnitte zu ermitteln (vgl. Definition von Werkzeug IV).

3.5 Ansätze für die Implementierung der Planungswerkzeuge

Im Folgenden werden Ansätze für die Auswahl von Algorithmen und Zielfunktionen zur Implementierung der Werkzeuge II bis V diskutiert. Die Umsetzung von Werkzeug I (Fixierung = Hinzufügen einer neuen Randbedingung) ist trivial und wird daher nicht näher betrachtet.

3.5.1 Implementierung von Werkzeug II

Zur Unterstützung der Wertauswahl für eine Aufgabe a muss das Planungssystem inkonsistente Werte identifizieren. Werte aus den Domänen der Startzeit- und Ressourcenvariablen, die bezüglich der ursprünglichen Randbedingungen inkonsistent sind, können für alle Aufgaben durch die Herstellung von k-Konsistenz oder lokaler Konsistenz vor dem interaktiven Planungsprozess ermittelt werden. Werte, die bezüglich der bisherigen Zuweisungen inkonsistent sind, können nach jedem Zuweisungsschritt durch die Herstellung lokaler Konsistenz (Propagierung) ermittelt werden. Bei der Betrachtung einer Aufgabe a sollte die Klassifizierung der Positionen im Plan (d.h. der Wertepaare $(start \in P, res \in RS)$) wie folgt ermittelt werden:

- Eine Position $(start \in P, res \in RS)$ ist inkonsistent bezüglich der ursprünglichen Randbedingungen, wenn der Wert $start$ und/oder der Wert res inkonsistent bezüglich der ursprünglichen Randbedingungen ist.
- Eine Position $(start \in P, res \in RS)$, die nicht inkonsistent bezüglich der ursprünglichen Randbedingungen ist, ist inkonsistent bezüglich der bisherigen Zuweisungen, wenn der Wert $start$ und/oder der Wert res inkonsistent bezüglich der bisherigen Zuweisungen ist.
- Eine Position $(start \in P, res \in RS)$ ist lokal konsistent bezüglich der bisherigen Zuweisungen, wenn sowohl $start$ als auch res lokal konsistent bezüglich der bisherigen Zuweisungen ist.

Wenn der Disponent für Aufgabe a eine konkrete Ressource auswählt, wird der Wertebereich der Startzeitvariablen a_{start} aufgrund der Randbedingungen für die-

se Ressource ggf. weiter eingeschränkt. In Ergänzung zu den oben beschriebenen Regeln können also ggf. weitere bezüglich der bisherigen Zuweisungen inkonsistente Positionen identifiziert werden, wenn alle Möglichkeiten der Ressourcenauswahl im Voraus betrachtet werden. Algorithmus 7 zeigt dafür eine mögliche Vorgehensweise.

Algorithmus 7 : Ermittlung zusätzlicher Positionen, die inkonsistent bezüglich der bisherigen Zuweisungen sind

input : Ein lokal konsistentes CSP (V, D, C), eine Aufgabe a mit $a_{start}, a_{res} \in V$

output : Eine Liste $Inkonsistent$ mit zusätzlichen inkonsistenten Positionen $(r \in dom_{a_{res}}, s \in dom_{a_{start}})$

```
1  Inkonsistent = {};
2  for each r ∈ dom_{a_res} do
3      /*Konsistenzherstellung nach Algorithmus 4:*/
4      CSP' ← local_consistency(CSP ∧ a_res = r);
5      if CSP' = false then
6          for each s ∈ dom_{a_start,CSP} do
7              Inkonsistent ← Inkonsistent ∪ (r, s);
8          end
9      else
10         for each s ∈ dom_{a_start,CSP} \ dom_{a_start,CSP'} do
11             /*Wenn s inkonsistent ist, füge (r,s) zur Liste hinzu:*/
12             Inkonsistent ← Inkonsistent ∪ (r, s);
13         end
14     end
15 end
```

Algorithmus 7 erhält als Eingabe ein lokal konsistentes CSP mit allen bisherigen Zuweisungen. Für jede potenzielle Ressource r einer Aufgabe a erfolgt eine Propagierung über das Constraint-Netzwerk des CSPs, um unter Annahme der Zuweisung dieser Ressource erneut lokale Konsistenz herzustellen. Die Prozedur *local_consistency* repräsentiert dabei ein Verfahren nach dem Vorbild von Algorithmus 4. Wenn die Konsistenzherstellung fehlschlägt, werden alle Paare (r, s) in die Liste der zusätzlichen inkonsistenten Werte aufgenommen. Nach einer erfolgreichen Konsistenzherstellung sind möglicherweise einige Werte aus der Domäne

von a_{start} entfernt worden. Für jeden Wert s, auf den dies zutrifft, wird das Paar (r, s) in die Liste der zusätzlichen inkonsistenten Positionen aufgenommen.

Diskussion: Einsatz von Filterfunktionen zur Ermittlung inkonsistenter Werte

Die Qualität der Wertauswahlunterstützung hängt davon ab, wie viele inkonsistente Werte von den einzelnen Filterfunktionen bei der Propagierung identifiziert werden. Es können ggf. Filterfunktionen eingesetzt werden, die bereits vom zugrunde liegenden Lösungsalgorithmus für die automatische Planerstellung angewendet werden (vgl. Abschnitt 2.3.6). Auch der Einsatz eines Constraintlösers, der nur die Propagierung im interaktiven Lösungsverfahren übernimmt und den bisherigen Planungsalgorithmus nicht ersetzt, bietet sich an. Bei professionellen Constraintlösern gehören Algorithmen zur Konsistenzherstellung zum Funktionsumfang (sodass z.B. für die Funktion *local_consistency*() keine Eigenimplementierung erforderlich ist).
Der Zugriff auf vorgefertigte Filterfunktionen ist jedoch u.U. nicht ausreichend, da sie nicht für den Zweck der Anwenderunterstützung entwickelt wurden, sondern lediglich eine Ergänzung für die Tiefensuche mit Backtracking darstellen. Für den Einsatz im interaktiven Lösungsalgorithmus muss die Domänenreduzierung so stark wie möglich sein, damit der Disponent so selten wie möglich mit Sackgassen konfrontiert wird. Der Einsatz von Propagierung zur Verbesserung der Anwenderinteraktion wurde von Dillan (2011) am Beispiel eines Flottenplanungsproblems untersucht. Seine Arbeit lässt folgende Rückschlüsse zu:

- Bei der Verwendung eines Constraintlösers sollte überprüft werden, ob das Planungsproblem mit *redundanten Randbedingungen* beschrieben werden kann (Rossi, van Beek und Walsh, 2006). Diese verändern den Lösungsraum des Problems nicht, aber sie beschreiben logische Zusammenhänge zwischen den Werten der Variablen, die in jeder Lösung gelten. Daraus können zusätzliche Regeln zur Filterung der Domänen bei der Herstellung lokaler Konsistenz gewonnen werden. Redundante Randbedingungen können in der Regel aus der Constraint-Bibliothek des Lösers ausgewählt werden, sodass die damit verknüpften Filteralgorithmen nicht selbst implementiert werden müssen.

- Für die vorliegenden Randbedingungen können eigene Filterfunktionen im-

plementiert werden, die bei der Herstellung lokaler Konsistenz die Reduktion der Domänen verstärken.

Die Rechenzeit, die dem Propagierungsverfahren zur Verfügung gestellt wird, sollte durch einen im System einstellbaren Timeout-Wert begrenzt werden. Bei der Überschreitung des Timeouts sollte die Propagierung abgebrochen werden. Damit wird die in Richtlinie 7 (Abschnitt 3.3.2) geforderte kurze Reaktionszeit des Systems sichergestellt. Konflikte, die bei der interaktiven Planung ggf. durch nicht identifizierte inkonsistente Werte verursacht werden, können vom Disponenten mit Hilfe der Planungswerkzeuge I bis V gelöst werden.

3.5.2 Implementierung von Werkzeug III

Werkzeug III berechnet einen Teilplan für einen gewählten Planabschnitt auf einer gewählten Abstraktionsebene. Im Folgenden gelten die Definitionen aus Abschnitt 2.1.

Eine Planungsfunktion für einen Abschnitt (I,RG) mit einem Intervall I und einer Ressourcengruppe RG löst ein Problem $CSOP = (V, D, C, f)$, wobei auf der Grundlage der allgemeinen Modellstruktur (Abschnitt 2.2) gilt: $V = \{a_{start} | a \in AS \wedge a_{start} \in I\} \cup \{a_{res} | a \in AS \wedge a_{res} \in RG\}$. Für einen Abschnitt (I,R) mit einer Ressource R gilt: $V = \{a_{start} | a \in AS \wedge a_{start} \in I\} \cup \{a_{res} | a \in AS \wedge a_{res} = R\}$. Die Menge D umfasst die Domänen der Variablen in V. Die Menge C wird mit den Randbedingungen initialisiert, die mindestens eine Variable $v \in V$ umfassen. Die Zielfunktion f wird aus dem CSP übernommen, welches das gesamte Planungsproblem beschreibt.

Vor der Planung und Optimierung müssen die Domänen D der in V enthaltenen Variablen initialisiert werden. Die konkreten Wertzuweisungen für Startzeiten und Ressourcen werden zurückgenommen und die Domänen werden anhand der Abschnittsbegrenzungen so eingeschränkt, dass gilt: $\forall a_{start}, a_{res} \in V : dom(a_{start}) \subseteq I, dom(a_{res}) \subseteq RG$ (bzw. $dom(a_{res}) = \{R\}$).

Zunächst sollte für $CSOP$ eine Konsistenzherstellung nach dem Vorbild von Algorithmus 4 durchgeführt werden. Wenn die Konsistenzherstellung scheitert, ist eine Lösung des Planungsproblems $CSOP$ für diesen Abschnitt nicht möglich (vgl. Konfliktfälle 1 - 4 bei Werkzeug III). Bei erfolgreicher Konsistenzherstellung kann versucht werden, eine Lösung für $CSOP$ zu finden. Dazu muss je nach Problem-

struktur ein geeignetes heuristisches oder exaktes Optimierungsverfahren gewählt werden (Abschnitt 2.3). Auch bei diesem Schritt kann sich noch herausstellen, dass das $CSOP$ nicht lösbar ist.
Die Rechenzeit für die Optimierung sollte wie bei Werkzeug II zeitlich begrenzt werden, um kurze Reaktionszeiten gem. Richtlinie 7 (Abschnitt 3.3.2) sicherzustellen. Wenn die Optimierung aufgrund einer Timeout-Überschreitung abgebrochen wurde, kann der Disponent seine Planungsentscheidungen ändern und versuchen, sie mit Hilfe der Planungswerkzeuge umzusetzen.

3.5.3 Implementierung von Werkzeug IV

Die bei Werkzeug IV unter Betrachtung einer Aufgabe a geschilderte Kategorisierung von Abschnitten baut auf der Kategorisierung einzelner Planpositionen durch Werkzeug II auf. Die Kategorien lassen sich nach folgendem Vorbild ermitteln:

Kategorie 1: Wenn innerhalb des betrachteten Abschnitts eine Position von Werkzeug II als konsistent klassifiziert wurde, wird der Abschnitt als konsistent markiert (Beispielfarbe Grün).

Kategorie 2: Wenn alle im Abschnitt enthaltenen Positionen von Werkzeug II als inkonsistent bezüglich der bisherigen Zuweisungen klassifiziert worden sind, muss intern[10] die dem Abschnitt zugeordnete Planungsfunktion aufgerufen werden. Wenn die Planungsfunktion einen Teilplan ermittelt, der unter Einbeziehung von Aufgabe a die bei Werkzeug III geschilderten Anforderungen erfüllt, kann der Abschnitt als konsistent vorbehaltlich der Anwendung einer Planungsfunktion markiert werden (Beispielfarbe Gelb).

Kategorie 3: Es gilt die gleiche Vorgehensweise wie bei Kategorie 2. Der Abschnitt wird in Kategorie 3 eingestuft, wenn die Planungsfunktion bei der Suche nach einem Teilplan unter Einbeziehung von Aufgabe a scheitert (Beispielfarbe Grau).

Kategorie 4: Wenn alle Positionen innerhalb des Abschnitts von Werkzeug II als

[10] Die probeweise Anwendung der Planungsfunktion ist nicht an der Benutzerschnittstelle sichtbar. Die intern vorgenommenen Änderungen am Plan werden nach der Anwendung der Planungsfunktion wieder rückgängig gemacht oder die Änderungen werden auf eine Kopie des Plans angewendet.

inkonsistent bezüglich der ursprünglichen Randbedingungen klassifiziert worden sind, wird der Abschnitt als inkonsistent bezüglich der ursprünglichen Randbedingungen markiert (Beispielfarbe Rot).

Zur Kennzeichnung aller Abschnitte auf einer bestimmten Abstraktionsebene sind demzufolge ggf. bereits vor der Zuweisung eines bestimmten Abschnitts zu Aufgabe a interne Aufrufe von Planungsfunktionen notwendig. Diese sollten in dem Moment erfolgen, in dem der Disponent die Absicht erkennen lässt, Aufgabe a im Plan zu verschieben bzw. in den Plan einzuordnen.
Für Abschnitte, die in Kategorie 3 eingestuft wurden, können ggf. weitere Informationen ermittelt werden, um die Entscheidungsunterstützung zu verbessern. Es können z.B. die Aufgaben im Abschnitt markiert werden, die zu einer (minimalen) Konfliktmenge von Variablen gehören (Rossi, van Beek und Walsh, 2006). Ein oder mehrere Aufgaben aus der Konfliktmenge müssen aus dem Abschnitt entfernt und in andere Abschnitte eingeordnet werden. Eventuell gehören nicht alle dem Abschnitt zugeordnete Aufgaben zur Konfliktmenge. Einige Aufgaben sind demzufolge nicht für das Scheitern der Planungsfunktion verantwortlich. Eine Entfernung dieser Aufgaben aus dem Abschnitt wäre somit wirkungslos, da die Planungsfunktion bei einem erneuten Aufruf wieder scheitern würde.
Ob Werkzeug IV bereitgestellt werden kann, hängt davon ab, ob die Vorberechnung innerhalb der vom Anwender tolerierten Wartezeit erfolgen kann. Wenn die maximale Wartezeit (Timeout) bei einem bestimmten Planabschnitt überschritten wird, sollte die Vorberechnung abgebrochen werden. In diesem Fall kann der Disponent die restlichen Werkzeuge für die Einplanung zur Hilfe nehmen. Je nach Problemstellung ist eine Umsetzung ggf. nur für bestimmte Abstraktionsebenen möglich. Außerdem ist es ggf. ausreichend, die Abschnitte zunächst nur in dem Teilbereich des Plans zu kennzeichnen, der vom Disponenten gerade betrachtet wird.

3.5.4 Implementierung von Werkzeug V

Werkzeug V ermittelt auf einer bestimmten Abstraktionsebene ae einen Plan L_{new}, der mit einem gegebenen Plan L_{old} weitestgehend übereinstimmt und gleichzeitig lokal konsistent ist. Der Grad der Übereinstimmung kann z.B. durch die in Gleichung 3.1 angegebene Zielfunktion bestimmt werden. Dabei bezeichnet a_{pos} eine

Position ($start \in P, res \in RS$) im Plan, wobei gilt: $a_{start} = start$ und $a_{res} = res$. Die Position, die a im Plan L_{old} bzw. L_{new} einnimmt, wird als $a_{pos_{old}}$ bzw. $a_{pos_{new}}$ bezeichnet.

$$\min \sum_{\forall a \in AS} dist(a_{pos_{old}}, a_{pos_{new}}, ae), ae \in \{0, ..., m-1\} \tag{3.1}$$

Die Funktion $dist : (P \times RS) \times (P \times RS) \times \{0, ..., m-1\} \rightarrow \mathcal{R}$ misst mit einer geeigneten Heuristik den Abstand des Planabschnitts, in dem sich die neue Position $a_{pos_{new}}$ befindet, vom Planabschnitt, in dem sich die alte Position $a_{pos_{old}}$ befindet. Der Abstand bezieht sich auf eine bestimmte Abstraktionsebene $ae \in \{0, ..., m-1\}$, m sei die Anzahl der Abstraktionsebenen. Bei $ae = 0$ wird der Abstand der neuen Position von der alten Position gemessen.
Mit dieser Zielfunktion wird eine Beibehaltung des ursprünglichen Abschnitts bzw. eine möglichst geringfügige Abweichung der neuen Abschnittszuordnung von der ursprünglichen Abschnittszuordnung für alle Aufgaben auf einer bestimmten Abstraktionsebene angestrebt. Je höher die Abstraktionsebene, desto größer sind die einzelnen Abschnitte. Demzufolge gilt: je höher die Abstraktionsebene, desto größere Abweichungen der konkreten Positionen sind zulässig, um einen minimalen Zielfunktionswert zu erreichen.
Werkzeug V sollte nur für die Abstraktionsebenen bereitgestellt werden, für die im durchschnittlichen Fall eine geringe Antwortzeit des Computers im interaktiven Planungsprozess realisiert werden kann (vgl. Abschnitt 3.3.3). Im schlechtesten Fall sollte die Berechnung ohne Ergebnis abgebrochen werden, sobald ein im System eingestelltes Timeout-Intervall abgelaufen ist. Der Disponent kann dann auf eine höhere Abstraktionsebene wechseln oder eine Anpassung des Plans mit Hilfe der restlichen Werkzeuge vornehmen.

3.6 Anwendungsspezifische Konfiguration der Planungswerkzeuge

Zur Implementierung der Werkzeuge III, IV und V wird ein Schema benötigt, nach dem die in Abbildung 3.2 gezeigte, allgemeine Planansicht in disjunkte Abschnitte unterteilt werden kann. Die Ansicht lässt dafür zunächst mehrere Alternativen offen. Sie kann z.B. in die vertikalen Abschnitte (UI1,RGS), (UI2,RGS) usw. unterteilt werden. Alternativ ist eine Unterteilung in horizontale Abschnitte, wie

z.B. in (I1, R1), (I1, R2) usw. möglich. Unter Umständen ist es erwünscht, dass der Disponent zwischen einer horizontalen und vertikalen Unterteilung wechseln kann. Dafür können an der Benutzerschnittstelle mehrere Planansichten bereitgestellt werden. Außerdem können mehrere Abstraktionsebenen existieren, die sich darin unterscheiden, auf welcher Hierarchieebene auf der Zeit- oder Ressourcenachse die Abschnittsbegrenzung einer Planungsfunktion (vgl. Abschnitt 3.4.4) erfolgt. So kann z.B. anstelle des Abschnitts (I1, R1) auch der Abschnitt (I1, RG1) die Grundlage einer Planungsfunktion auf einer höheren Abstraktionsebene bilden. Ein Wechsel zwischen den Abstraktionsebenen kann ein Bestandteil des interaktiven Planungsprozesses sein, um das Ausmaß der Umplanung zu kontrollieren.
Bei der Gestaltung von Planungssystemen muss die gewählte Anzahl von Abstraktionsebenen und die Abschnittsunterteilung auf den einzelnen Abstraktionsebenen dem Problemverständnis des Disponenten und seiner typischen Vorgehensweise bei der Planung entsprechen. Vor der Implementierung eines Planungssystems sollte daher eine Analyse stattfinden, bei der die in dem jeweiligen Anwendungsfall typischen Entscheidungsvorgänge ermittelt werden. In den folgenden Abschnitten wird das Konzept der *Zuweisungsregel* vorgestellt, welches genutzt werden kann, um eine Vorlage für die Anordnung der Ebenen und Abschnitte zu ermitteln.

3.6.1 Zuweisungsregeln

Bei der manuellen Planung werden nacheinander bestimmte Eigenschaften oder *Attribute* einer Aufgabe a betrachtet. Ein Attribut entspricht dabei einer bestimmten Hierarchieebene der Unterteilung der Zeit- oder Ressourcenachse. Die Wertzuweisung für ein Attribut erfolgt durch die Auswahl eines bestimmten Abschnitts auf der jeweiligen Ebene, womit gleichzeitig die Eingrenzung des Wertebereiches von a_{start} bzw. a_{res} auf den jeweiligen Teilbereich des Planungshorizontes bzw. auf die jeweilige Ressourcengruppe oder Ressource verbunden ist.
Häufig werden die Attribute einer Aufgabe stets in einer bestimmten Reihenfolge betrachtet. Nach dem Schema in Abbildung 3.2 ist z.B. die Zuweisungsreihenfolge $I \rightarrow RG \rightarrow R \rightarrow UI \rightarrow Startzeit$ möglich, mit $I = \{I1, I2\}, UI = \{UI1, ..., UI6\}$, $RG = \{RG1, RG2\}$ und $R = \{R1, ..., R4\}$. Darüber hinaus wird vor der Festlegung der konkreten Startzeiten von Aufgaben häufig zunächst deren Reihenfolge festgelegt. Die Reihenfolge kann daher ggf. ebenfalls als Attribut aufgefasst werden.

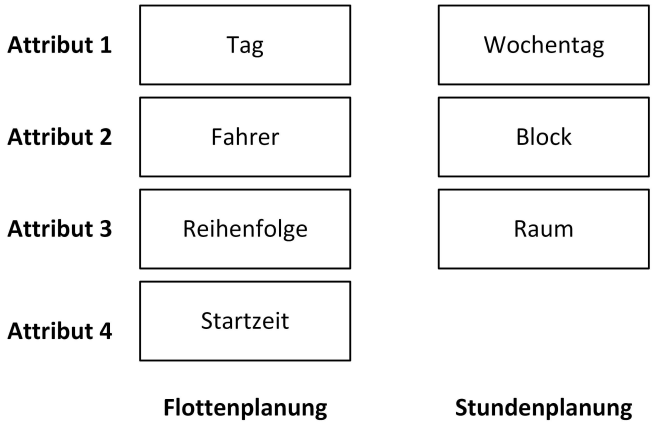


Abbildung 3.8: Typische Zuweisungsregeln für Planungsprobleme

Die Vorschrift, durch welche die Zuweisungsreihenfolge der Attribute für jede Aufgabe a des Planungsproblems festgelegt wird, soll als *Zuweisungsregel* bezeichnet werden. Für ein bestimmtes Planungsproblem können auch mehrere zulässige Zuweisungsregeln aufgestellt werden.
Die Zuweisungsregel steht häufig im Zusammenhang mit einer Lösungsstrategie nach dem Prinzip „Teilen und Herrschen“. Dabei ist es das Ziel, Teilprobleme zu bilden, die weitestgehend unabhängig voneinander gelöst werden können. Zuweisungen, die zur Bildung von Teilproblemen dienen, werden daher eher vorgenommen, als Zuweisungen, die zur Lösung der jeweiligen Teilprobleme erforderlich sind. Ein typisches Beispiel sind Verfahren zur Flottenplanung (Beispiel 3.5).

Beispiel 3.5 (Flottenplanung): Abbildung 3.8 zeigt eine Zuweisungsregel für die Flottenplanung. Die Fahrer beginnen und beenden ihre Touren an jedem Arbeitstag im Depot des Unternehmens. Der Disponent ordnet die Aufgaben daher zunächst einem bestimmten Arbeitstag zu (Attribut 1). Anschließend nimmt er für jeden Arbeitstag die Verteilung der Aufgaben auf die Fahrer vor (Attribut 2). Für jeden Fahrer werden die Abfolge der Touren (Attribut 3) und die Startzeiten der Aufgaben (Attribut 4) festgelegt.

Auch im folgenden Beispiel werden Teilprobleme gebildet, die weitestgehend unabhängig voneinander gelöst werden können. In der Stundenplanung können Kurse, die bereits unterschiedlichen Wochentagen zugeordnet sind, in der Regel unabhängig voneinander einem konkreten Block zugeteilt werden, d.h. die einzelnen

Wochentage bilden Teilprobleme (ein entsprechendes Planungsmodell ist bei Anh, Tam und Hung (2006) zu finden).

Beispiel 3.6 (Universitätsstundenplanung): Abbildung 3.8 zeigt eine Zuweisungsregel für die Universitätsstundenplanung. Die Kurse werden zuerst auf die Wochentage verteilt (Attribut 1) und anschließend einem bestimmten Block[11] zugeordnet (Attribut 2). Die Raumzuweisung wird den zeitlichen Attributen untergeordnet, da sie die Zufriedenheit von Studenten und Dozenten mit der Planung in der Regel am wenigsten beeinflusst.

Die Stundenplanung ist jedoch ein Anwendungsfall, bei dem nicht für alle Aufgaben von der gleichen Zuweisungsregel ausgegangen werden kann. Für bestimmte Kurse kann die Raumzuweisung, z.B. aufgrund einer bestimmten Ausstattung oder eines Buchungswunsches, eine höhere Priorität besitzen als die Zeit. Für sie liegen die Raumzuweisungen daher auf einer höheren Ebene als für die restlichen Kurse.
Die vorliegenden Beispiele zeigen, dass Zuweisungsregeln nicht nur dem typischen Problemverständnis des Disponenten entsprechen, sondern auch eine Vorlage für eine geeignete Implementierung der Planungswerkzeuge darstellen, mit der eine effiziente Computerunterstützung geleistet werden kann.

3.6.2 Gestaltung der Fixierungsmöglichkeiten aufgrund einer Zuweisungsregel

Die von einem Planungssystem angebotenen Fixierungsmöglichkeiten sollten sich an der zugrunde liegenden Zuweisungsregel orientieren. Bei n Attributen muss die Fixierung eines Abschnitts für Attribut $j, 1 \leq j \leq n$ für eine bestimmte Aufgabe nach oben verkettet werden, d.h. die (übergeordneten) Abschnitte für die Attribute 1 bis $j-1$ müssen für diese Aufgabe ebenfalls fixiert werden. Eine Fixierung für Attribut n entspricht der Festlegung einer konkreten Position ($start \in P, res \in RS$). Für ein Flottenplanungsproblem mit der in Abbildung 3.8 gezeigten Zuweisungsregel heißt das z.B.: Bei der Fixierung einer Ressource (Attribut 2) wird der vorher ausgewählte Arbeitstag für diese Aufgabe mitfixiert. Wenn mehrere zulässige Zu-

[11] Begriffsklärung Block: Kurse finden i.d.R. in festen Zeitintervallen statt, wie z.B. Block 1 = 8:00-9:30 Uhr, Block 2 = 10:00-11:30 Uhr

weisungsregeln existieren, können mehrere Planansichten mit unterschiedlichen Fixierungsmöglichkeiten bereitgestellt werden. Wenn keine Zuweisungsregel existiert, sollte die Fixierung (unter Beachtung der Hierarchie der Zeit- bzw. Ressourcenabschnitte) unabhängig von einer bestimmten Reihenfolge ermöglicht werden.

3.6.3 Gestaltung der Werkzeuge III bis V aufgrund einer Zuweisungsregel

Eine Zuweisungsregel liefert eine Vorlage für die Gestaltung von Abstraktionsebenen und Planungsfunktionen. Eine Zuweisung eines Abschnitts für Attribut $j, j < n$ (n sei die Anzahl der Attribute) kann jeweils auf einer höheren Abstraktionsebene ae angesiedelt werden als die nachfolgende Zuweisung für Attribut $j+1$. Die Anzahl der benötigten Abstraktionsebenen ergibt sich aus $n+1$ (eine Ebene pro Attribut + eine zusätzliche Ebene für den Gesamtplan). Die Abstraktionsebenen können z.B. nach folgenden Prinzip aus einer Zuweisungsregel abgeleitet werden:

1. *Initialisierung:*
   - $ae \leftarrow n$
   - $j \leftarrow 0$
   - Führe Abstraktionsebene ae ein. Auf dieser wird der Gesamtplan (P, RGS) betrachtet.
2. *Wiederhole, bis* $ae = 0 \wedge j = n$*:*
   a) $ae \leftarrow ae - 1$
   b) $j \leftarrow j + 1$
   c) Führe Abstraktionsebene ae ein. Hier werden (Abschnitts-)Zuweisungen für das Attribut j betrachtet.

Die Unterteilung in Abschnitte und die Zuordnung entsprechender Planungsfunktionen kann auf jeder Abstraktionsebene ae nach einer der folgender Vorlagen vorgenommen werden:

- *Vorlage A:* Unterteile den Plan (für $ae = n$) bzw. jeden Abschnitt des Plans (für $ae < n$) in (horizontale oder vertikale) (Teil-)Abschnitte, die durch das Attribut mit der Nummer j definiert werden. Für jeden Abschnitt wird eine Planungsfunktion entwickelt, welche die in Abschnitt 3.4 definierten Anforderungen erfüllt. Diese ermitteln für alle Aufgaben innerhalb des Abschnitts die Wertzuweisungen für die Attribute mit den Nummern $j+1$ bis n (auf den Ebenen $ae-1$ bis 0) automatisch und berücksichtigen die ggf. vorhandenen Fixierungen.
  Ausnahmen: Für das Attribut „Reihenfolge“ wird die Abschnittsunterteilung der nächst höheren Abstraktionsebene übernommen. Es wird eine Planungsfunktion entwickelt, welche für die Aufgaben innerhalb eines Abschnitts die Einhaltung der vom Disponenten vorgegebenen Reihenfolge sicherstellt. Für das Attribut auf der untersten Abstraktionsebene 0 (z.B. Attribut „Startzeit“) wird keine Planungsfunktion entwickelt.

- *Vorlage B:* Unterteile die Abschnitte wie in Vorlage A. Es wird eine Planungsfunktion entwickelt, die alle Planungsfunktionen der Teilabschnitte gleichzeitig aktiviert.

- *Vorlage C:* Unterteile die Abschnitte wie in Vorlage A. Für jeden Teilabschnitt wird eine Planungsfunktion entwickelt, die eine vom Nutzer frei wählbare Teilmenge von Aufgaben innerhalb des Abschnitts neu einplant. Die im Abschnitt befindlichen, nicht ausgewählten Aufgaben werden auf der untersten Abstraktionsebene 0 (Attribut n) automatisch fixiert. Das bedeutet, dass ihre Position von der Planungsfunktion nicht verschoben werden darf.

- *Vorlage D:* Wie Vorlage C, aber die im Abschnitt befindlichen, nicht ausgewählten Aufgaben werden auf einer bestimmten, niedrigeren Abstraktionsebene zwischen $ae-1$ und 0 (d.h. für ein Attribut zwischen $j+1$ und n) automatisch fixiert. Das bedeutet, dass sie innerhalb des Teilabschnitts, dem sie auf dieser Abstraktionsebene zugeordnet sind, verschoben werden dürfen.

Die Vorlagen C und D bieten sich u.a. zur Konfliktlösung an (vgl. Konfliktursachen in Abschnitt 3.4.4). Die am Konflikt beteiligten Aufgaben können auf einer bestimmten Abstraktionsebene ausgewählt werden. Durch die Anwendung der entsprechenden Planungsfunktion wird ihre Position korrigiert, wobei die übrigen Aufgaben nur bis zu einem gewissen Grad beweglich sind.

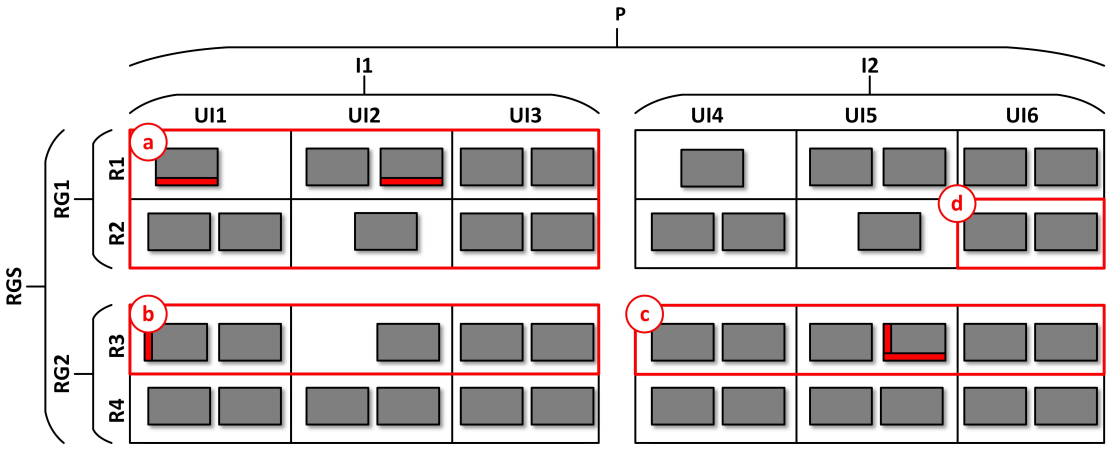


Abbildung 3.9: Schema eines Plans mit 4 ausgewählten Planungsfunktionen a), b), c) und d) auf den Abstraktionsebenen 1, 2 und 3 (Beispiel 3.7).

Beispiel 3.7 zeigt die Anwendung von Vorlage A für das allgemeine Planschema aus Abbildung 3.2.

Beispiel 3.7: Abbildung 3.9 stellt exemplarisch 4 Planungsfunktionen dar, die nach der Zuweisungsregel I (Attribut 1) $\rightarrow$ RG (Attribut 2) $\rightarrow$ R (Attribut 3) $\rightarrow UI$ (Attribut 4) $\rightarrow$ *Startzeit* (Attribut 5) nach Vorlage A ermittelt wurden:

- a) Planungsfunktion auf $ae = 3$ für alle Aufgaben innerhalb des Abschnitts (I1, RG1)
- b) und c) Planungsfunktionen auf $ae = 2$ für alle Aufgaben innerhalb von (I1, R3) bzw. (I2, R3)
- d) Planungsfunktion auf $ae = 1$ für alle Aufgaben innerhalb von (UI6, R2)

Fixierte Zuweisungen bleiben unverändert. Sie werden durch eine rote Aufgabenmarkierung gekennzeichnet: eine horizontale Markierung entspricht der Fixierung der Ressource, eine vertikale Markierung der Fixierung der Zeit.

In den Beispielen 3.8 und 3.9 werden Planungsfunktionen für ein Flottenplanungs- und ein Stundenplanungsproblem vorgestellt, die nach dem oben beschriebenen Prinzip ermittelt wurden (vgl. Beispiel 3.5 und 3.6).

Beispiel 3.8 (Flottenplanung): Eine Zuweisungsregel für ein Flottenplanungsproblem ist in Abbildung 3.8 dargestellt. Es wurden folgende Abstraktionsebenen identifiziert:

$ae = 4$**:** Hier wird der Gesamtplan betrachtet.

$ae = 3$**:** Es erfolgt eine Unterteilung des Gesamtplans, sodass für jeden Tag ein Teilabschnitt vorliegt, der alle Ressourcen beinhaltet.

$ae = 2$**:** Die Abschnitte von $ae = 3$ werden weiter unterteilt, sodass für jede Ressource ein Teilabschnitt vorliegt.

$ae = 1$**:** Es erfolgt die gleiche Abschnittsunterteilung wie bei $ae = 2$. Diese Ebene wird zur Berücksichtigung von Reihenfolgepräferenzen benötigt (vgl. Vorlage A).

$ae = 0$**:** Die unterste Abstraktionsebene beinhaltet die einzelnen Startzeiten der Aufgaben. Hier ist für die vorliegende Zuweisungsregel keine weitere Abschnittsunterteilung möglich (vgl. Vorlage A).

Daraus können die nachfolgenden Planungsfunktionen abgeleitet werden (Abbildung 3.10). Jede Planungsfunktion strebt stets eine Minimierung der Fahrzeiten für die einzelnen Touren und einen Ausgleich der Arbeitszeiten zwischen allen Fahrern an. Die vom Disponenten vorgenommene Gewichtung der Zielfunktionen wird dabei berücksichtigt.

Full Optimization (FO): Es erfolgt für alle Aufgaben die automatische Ermittlung aller Zuweisungen für Attribut 1 bis 4 ($ae = 4$).

Day Optimization (DO1 und DO2): Automatische Ermittlung der Zuweisungen von Attribut 2 bis 4. Die Planungsfunktion DO1 wird jeweils auf die Aufgaben angewendet, die einem bestimmten Tag zugeordnet sind (Vorlage A). DO2 wendet DO1 für alle Tage gleichzeitig an ($ae = 3$, Vorlage B).

Group Optimization 1 (GO1): Es erfolgt für ausgewählte Aufgaben, die einem bestimmten Tag (Attribut 1) zugeordnet sind, die automatische Ermittlung aller Zuweisungen für Attribut 2 bis 4 ($ae = 3$, Vorlage C).

Group Optimization 2 (GO2): Diese Funktion entspricht GO1. Eine automati-

sche Fixierung der im Abschnitt befindlichen, nicht ausgewählten Aufgaben erfolgt auf Abstraktionsebene 4 bzw. für die Attribute 1 bis 3 ($ae = 3$, Vorlage D). Die Startzeiten (Attribut 4) der nicht ausgewählten Aufgaben dürfen somit von der Planungsfunktion verändert werden, aber die ursprüngliche Reihenfolge (Attribut 3) muss beibehalten werden.

Resource Optimization (RO1 und RO2): Automatische Ermittlung der Zuweisungen von Attribut 3 bis 4 für alle Aufgaben. Die Planungsfunktion RO1 wird jeweils auf die Aufgaben angewendet, die einer bestimmten Ressource an einem bestimmten Tag zugeordnet sind (Tourenoptimierung, Vorlage A). RO2 wendet RO1 für alle Ressourcen gleichzeitig an ($ae = 2$, Vorlage B).

FIT-IN1 und 2: Automatische Ermittlung der Zuweisungen für Attribut 4. FIT-IN1 wird jeweils auf die Aufgaben angewendet, die einer bestimmten Ressource an einem bestimmten Tag zugeordnet sind und für die eine bestimmte Reihenfolge festgelegt wurde (Aufheben von zeitlichen Überlappungen, Vorlage A). FIT-IN2 wendet FIT-IN1 für alle Ressourcen gleichzeitig an ($ae = 1$, Vorlage B).

Ein Teil dieser Planungsfunktionen wurde bereits in (Prenzel, 2011) veröffentlicht.

Bei einigen Planungsproblemen müssen Planungsfunktionen für mehrere typische Zuweisungsregeln bereit gestellt werden. Oft ist es sinnvoll, die Entscheidung, nach welcher Zuweisungsregel eine bestimmte Aufgabe oder Aufgabengruppe eingeplant werden soll, dem Disponenten zu überlassen. In Beispiel 3.9 (Stundenplanung, vgl. Abschnitt 2.2.3 und Abbildung 3.8) werden zwei Zuweisungsregeln mit der jeweiligen Attributabfolge (3,1,2) und (1,2,3) berücksichtigt. Bei der Anwendung der Planungsfunktionen RS und DS wird erstere zugrunde gelegt, bei der Anwendung von DS2 und BS letztere. Die Funktion FS ist für beide Zuweisungsregeln äquivalent, da die Abfolge, in der die Attribute 1 bis 3 vom Computer verarbeitet werden, von den Zuweisungsregeln abweichen kann. Die Verarbeitung kann als „Black Box“ betrachtet werden.

Beispiel 3.9 (Universitätsstundenplanung): Eine Zuweisungsregel für ein Stundenplanungsproblem ist in Abbildung 3.8 dargestellt. Für die Zuweisungsregel

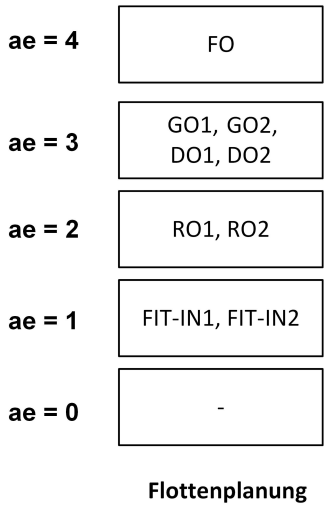


Abbildung 3.10: Planungsfunktionen, die sich aus der Zuweisungsregel eines Flottenplanungsproblems ergeben.

(3,1,2) wurden folgende Abstraktionsebenen identifiziert:

$ae = 3$**:** Hier wird der Gesamtplan betrachtet.

$ae = 2$**:** Es erfolgt eine Unterteilung des Gesamtplans, sodass für jeden Raum ein Teilabschnitt vorliegt, der alle Tage beinhaltet.

$ae = 1$**:** Die Abschnitte von $ae = 2$ werden weiter unterteilt, sodass für jeden Tag ein Teilabschnitt vorliegt.

$ae = 0$**:** Die unterste Abstraktionsebene beinhaltet die einzelnen Blöcke. Hier ist für die vorliegende Zuweisungsregel keine weitere Abschnittsunterteilung möglich (vgl. Vorlage A).

Daraus können nach Vorlage A folgende Planungsfunktionen abgeleitet werden (Abbildung 3.11):

Full Scheduling (FS): Automatische Ermittlung aller Zuweisungen von Attribut 1 bis 3 für alle Kurse ($ae = 3$).

Room Scheduling (RS): Automatische Ermittlung der Zuweisungen von Attribut 1 bis 2 für alle Kurse, die einem bestimmten Raum zugeordnet sind ($ae = 2$).

Day Scheduling (DS): Automatische Ermittlung der Zuweisungen von Attribut 2 für alle Kurse, die einem bestimmten Tag und Raum zugeordnet sind ($ae = 1$).

Für die Zuweisungsregel (1,2,3) wurden folgende Abstraktionsebenen identifiziert:

$ae = 3$**:** Hier wird der Gesamtplan betrachtet.

$ae = 2$**:** Es erfolgt eine Unterteilung des Gesamtplans, sodass für jeden Tag ein Teilabschnitt vorliegt, der alle Blöcke und Räume beinhaltet.

$ae = 1$**:** Die Abschnitte von $ae = 2$ werden weiter unterteilt, sodass für jeden Block ein Teilabschnitt vorliegt, der alle Räume beinhaltet.

$ae = 0$**:** Die unterste Abstraktionsebene beinhaltet die einzelnen Räume. Hier ist für die vorliegende Zuweisungsregel keine weitere Abschnittsunterteilung möglich (vgl. Vorlage A).

Daraus können nach Vorlage A folgende Planungsfunktionen abgeleitet werden (Abbildung 3.11):

Full Scheduling (FS): Automatische Ermittlung aller Zuweisungen von Attribut 1 bis 3 für alle Kurse ($ae = 3$).

Day Scheduling 2 (DS2): Automatische Ermittlung der Zuweisungen von Attribut 2 bis 3 für alle Kurse, die einem bestimmten Tag zugeordnet sind ($ae = 2$).

Block Scheduling (BS): Automatische Ermittlung der Zuweisungen von Attribut 3 für alle Kurse, die einem bestimmten Tag und Block zugeordnet sind ($ae = 1$).

Bei der Gestaltung der Werkzeuge zur Unterstützung der Wertauswahl (IV) und zur Konfliktauflösung (V) sollten die gleichen Abstraktionsebenen und Abschnitte zugrunde gelegt werden, die bei der Ermittlung der Planungsfunktionen aus der Zuweisungsregel abgeleitet wurden. Wenn die Reihenfolge Bestandteil der Zuweisungsregel ist, sollte bei der Konfliktauflösung auf der zugehörigen Abstraktionsebene eine minimale Abweichung von der Reihenfolge gesucht werden. Wenn die

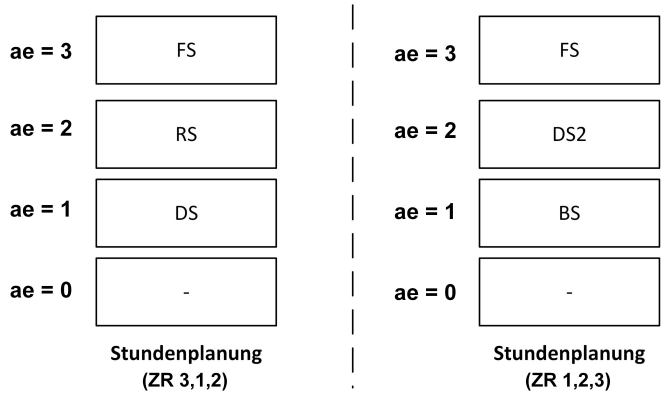


Abbildung 3.11: Planungsfunktionen, die sich aus zwei verschiedenen Zuweisungsregeln (ZR) für ein Stundenplanungsproblem ergeben.

Einhaltung der Reihenfolge nicht möglich ist, wird auf höhere Ebenen gewechselt.

3.7 Anwendung der Planungswerkzeuge im integrierten Lösungsprozess

In Abschnitt 3.1.3 wurde der wissensbasierte Planungsprozess des Disponenten modelliert (Algorithmen 5 und 6). Daraus konnten Defizite in der Mensch-Computer-Interaktion abgeleitet werden (Abschnitt 3.2). Mit Hilfe der Interaktionswerkzeuge I bis V können diese Defizite behoben werden. Im Folgenden wird gezeigt, wie sich die wissensbasierte Planung verändert, wenn die Werkzeuge zur Verfügung stehen. Anschließend wird untersucht, welche Verbesserungen sich aus dem neuen Prozess ergeben. Bevor im letzten Abschnitt ein Beispiel angegeben wird, erfolgt eine Abgrenzung der Planungswerkzeuge zu den Richtlinien von Frayman (2001).

3.7.1 Ein neuer interaktiver Algorithmus

Die Werkzeuge I bis V beeinflussen den Interaktionsprozess zur Erstellung einzelner, ggf. partieller Alternativlösungen im Rahmen eines wissensbasierten Entscheidungsprozesses (Abschnitt 2.4.3). Algorithmus 5 bleibt als Rahmen des Entscheidungsprozesses zur iterativen Vervollständigung und Auswertung von Alternativlösungen erhalten. Algorithmus 6 muss zur Berücksichtigung der Werkzeuge neu

formuliert werden. Der Interaktionsprozess, der vorher als allgemeiner Lösungsvorgang für CSPs modelliert wurde, soll dabei auf Planungsprobleme spezialisiert werden, um die Einbindung der Planungswerkzeuge zu ermöglichen.
Der neue Interaktionsprozess wird in Algorithmus 8 gezeigt. Jede Variable entspricht hier einer Aufgabe, für die auf Abstraktionsebene 0 eine Position im Plan gesucht wird, d.h. $|V| = |AS|$ (es gelten die Definitionen aus Abschnitt 2.2). Die Menge der Positionen auf Ebene 0 sei $Pos = P \times RS$. Für jede Variable $v \in V$, die zu Aufgabe $a \in AS$ gehört, gilt daher: $dom(v) = \{(s,r)|(s,r) \in Pos \wedge s \in dom(a_{start}) \wedge r \in dom(a_{res})\}$.
Der neue Interaktionsprozess sieht eine Planung auf mehreren Abstraktionsebenen vor. Auf einer Ebene $ae, ae > 0$ entspricht eine Position einem Planabschnitt (I, RG), der sich aus einem Abschnitt I auf der Zeitachse und einem Abschnitt RG auf der Ressourcenachse zusammensetzt (vgl. Abbildung 3.2). Bei der Zuweisung einer Aufgabe a zu einer Position auf einer Ebene $ae, ae > 0$ wird die Domäne der zugehörigen Variable v auf die vom Planabschnitt umfassten Positionen $p, p \in Pos$ begrenzt. Algorithmus 8 ist wie folgt aufgebaut:

Zeile 1: Die Listen *scheduled* und *unscheduled* enthalten jeweils die bereits eingeplanten bzw. die noch nicht eingeplanten Aufgaben des Planungsproblems. Sie bilden die Ausgangsbasis für die Erstellung oder Modifizierung einer Alternativlösung.

Zeile 3: Der Disponent kann einzelne Aufgaben aus der Liste *scheduled* auf beliebigen Abstraktionsebenen fixieren oder vorhandene Fixierungen auflösen (Werkzeug I).

Zeile 5: Zunächst erfolgt die Wahl einer bestimmten Abstraktionsebene. Auf dieser Ebene wird für die im Folgenden ausgewählte Aufgabe eine Position gesucht.

Zeilen 6 bis 10: Der Disponent wählt eine beliebige Aufgabe aus, deren Position geändert bzw. neu zugewiesen werden soll. Eine zugewiesene Aufgabe wird vom Disponenten sofort in die Liste noch nicht zugewiesener Aufgaben verschoben (Zeilen 8-9). Algorithmus 8 ist in diesem Fall abgeschlossen und der Disponent kann sich anschließend in Algorithmus 5 (Zeile 13) entscheiden, ob die Suche nach einer Position für diese Aufgabe in der nächsten Iteration manuell ($userComplete = false$) oder in dieser Iteration vom Computer

($userComplete = true$) vorgenommen werden soll.

Zeilen 11 bis 18: Eine ausgewählte Aufgabe, die nicht zugewiesen ist, wird zuerst aus der Liste *unscheduled* entfernt (Zeile 11). Die Auswahl einer Position für die gewählte Aufgabe wird im weiteren Verlauf durch die **Werkzeuge II und IV** unterstützt. Dabei ermittelt die Interaktionsfunktion *computerShowPositions*() konsistente Zuweisungen und kennzeichnet inkonsistente Zuweisungen auf der gewählten Abstraktionsebene nach dem Aufwand der notwendigen Umplanung (Zeile 12, vgl. Abbildung 3.6). Der Disponent kann eine oder einen Teil der vorgeschlagenen Positionen für eine nähere Betrachtung auswählen[12] (Zeile 13). In den Zeilen 14 bis 18 werden die ausgewählten Positionen[13] nacheinander evaluiert (Prozedur *userEvaluate*()). Daraus ergeben sich die Kandidaten, die aus Sicht des Disponenten für die tatsächliche Zuweisung in Frage kommen.

Zeilen 19 bis 29: Wenn keine kandidierende Position vorliegt, wird die Aufgabe ausgeplant (Zeile 20). Ansonsten wählt der Disponent eine kandidierende Position aus und weist sie der Aufgabe zu[14] (Zeile 22). Wenn die Position inkonsistent bezüglich der bisherigen Zuweisungen ist, muss gleichzeitig eine manuelle oder automatische Umplanung ausgeführt werden. Dafür können die **Werkzeuge III und V** eingesetzt werden. Welche Umplanungsschritte notwendig sind, wurde bereits bei der Evaluierung möglicher Positionen ermittelt (Zeile 17 bzw. Algorithmus 9). Der Vorgang wird durch die Prozedur *userCompleteAssignment* repräsentiert (Zeile 23). Wenn der Disponent auf einer höheren Abstraktionsebene arbeitet ($ae > 0$), wurde für die Aufgabe noch keine konkrete Position ausgewählt. Daher wird sie zurück in die Liste *unscheduled* verschoben (Zeile 27).

Zeile 30: Der Teilprozess zum Zurücknehmen oder Ändern einer Position einer Aufgabe ist nun abgeschlossen. Die Interaktion wird nun in Algorithmus 5,

[12] Der Disponent muss die ausgewählten Positionen nicht an der Benutzerschnittstelle kennzeichnen. Es soll jedoch verdeutlicht werden, dass u.U. nur ein Teil der vom Computer vorgeschlagenen Positionen für den Disponenten interessant ist.

[13] Es wird davon ausgegangen, dass der Disponent keine Position auswählt, die inkonsistent bezüglich der ursprünglichen Randbedingungen ist.

[14] Diese Zuweisung gilt (im Gegensatz zu einer mit Werkzeug I fixierten Zuweisung) nicht als harte Randbedingung und darf bei der automatischen Planung (Algorithmus 5, Zeile 6, 15) bzw. von den Werkzeugen III und V modifiziert werden.

Zeile 13 fortgesetzt (vgl. Abschnitt 3.1.3).

Algorithmus 9 beschreibt den Vorgang zur Bewertung einer vom Computer vorgeschlagenen Position. Die Position wird zunächst probeweise zugewiesen (Zeile 3). Die Prozedur $computerIsConsistent()$ gibt den Wert $false$ zurück, wenn die Position in Algorithmus 8, Zeile 12 als inkonsistent klassifiziert wurde (Zeile 4). Mit Hilfe der Werkzeuge III und V kann der Disponent versuchen, die Position für die Aufgabe *task* konsistent zu machen. Um zu verhindern, dass die ausgewählte Position von den Werkzeugen verändert wird, muss der Disponent die Aufgabe zunächst an dieser Position fixieren (Zeile 6). Die Prozedur $reschedule()$ (Zeile 8) wird wie folgt interpretiert: Der Disponent ruft die Planungsfunktion der aktuellen Abstraktionsebene für den Abschnitt *position* auf (Werkzeug III). Wenn diese Umplanung scheitert, so ruft der Disponent die durch Werkzeug V bereitgestellte Planungsfunktion auf, um auf der aktuellen Abstraktionsebene eine abschnittsübergreifende Umplanung durchzuführen. Wenn die Umplanung durch Werkzeug III oder V also erfolgreich ist, gibt die Prozedur *reschedule* den Wert *true* zurück, ansonsten $false$.
Wenn die Umplanung nicht erfolgreich war, wird die Positionierung der Aufgabe zurückgenommen und die Prozedur $userEvaluate()$ anschließend beendet (Zeilen 9 und 10). Im Erfolgsfall (die Position war konsistent oder wurde durch die Umplanung konsistent gemacht), wird die bewertete Position in die Liste kandidierender Positionen übernommen, wenn der Disponent mit dem Ergebnis zufrieden ist (Zeilen 13 und 14 und Algorithmus 8, Zeile 17). Bevor die Prozedur $userEvaluate$ endet, werden die ggf. vorgenommenen Planänderungen und die Positionierung der Aufgabe zurückgenommen (Zeilen 16 bis 19).

Algorithmus 8 : Interaktive Planung mit den Werkzeugen I bis V

```
1  userSolve(scheduled, unscheduled) ≡
2  /*ggf. Anwendung von Werkzeug I:*/
3  userFixUnfix(scheduled);
4  /*Auswahl der Abstraktionsebene:*/
5  ae ← userSelectLevel();
6  /*Auswahl einer Aufgabe aus scheduled oder unscheduled:*/
     v_select ← userSelectTask(scheduled, unscheduled);
7  if v_select ∈ scheduled then
8      scheduled ← scheduled\{v_select};
9      unscheduled ← unscheduled ∪ {v_select};
10 else
11     unscheduled ← unscheduled\{v_select};
12     /*Unterstützung der Positionsauswahl durch Werkzeuge II und IV:*/
         suggestedPostions ←
         computerShowPositions(ae, v_select, scheduled, unscheduled);
13     positions ← userSelectPositions(suggestedPositions);
14     candidates ← ∅;
15     /*Wahl von Kandidaten, die zulässig und zufriedenstellend sind:*/
16     for each position ∈ positions do
17         candidates ←
             candidates ∪ {userEvaluate(ae, v_select, position, scheduled, unscheduled)};
18     end
19     if candidates = ∅ then
20         unscheduled ← unscheduled ∪ {v_select};
21     else
22         position ← userSelectPosition(candidates);
23         userCompleteAssignment(ae, v_select, position);
24         if ae = 0 then
25             scheduled ← scheduled ∪ {v_select};
26         else
27             unscheduled ← unscheduled ∪ {v_select};
28         end
29     end
30     return;
31 end
```

Algorithmus 9 : Probeweise Umsetzung und Bewertung der Einplanung einer Aufgabe an eine bestimmte Position im Plan

```
1  userEvaluate(ae, task, position, scheduled, unscheduled) ≡
2  selectedPosition ← ∅;
3  userAssign(task, position);
4  if !computerIsConsistent?(ae, task, position) then
5  |  /*Werkzeug I:*/
6  |  userFix(task, position);
7  |  /*Werkzeuge III und V:*/
8  |  if !reschedule(ae, scheduled, unscheduled) then
9  |  |  userAssign(task, null);
10 |  |  return selectedPosition;
11 |  end
12 end
13 if userIsSatisfying?(scheduled, unscheduled) then
14 |  selectedPosition ← position;
15 end
16 /*Automatische Planänderungen rückgängig machen:*/
17 undo();
18 userAssign(task, null);
19 return selectedPosition;
```

Wenn im Rahmen der Prozedur *userEvaluate*() die Zuweisung einer Position für Aufgabe a nicht zufriedenstellend auf Abstraktionsebene ae realisiert werden konnte (es gilt $selectedPosition = \emptyset$), stehen dem Disponenten folgende Möglichkeiten zur Verfügung:

- Er kann in Algorithmus 8 andere Positionen bewerten und daraus eine Auswahl treffen.
- Er kann Algorithmus 8 beenden, ohne eine Auswahl zu treffen. In einer neuen Iteration (vgl. Algorithmus 5) kann der Disponent anschließend
  - eine höhere Abstraktionsebene auswählen, um den Handlungsspielraum für die automatische Umplanung zu vergrößern oder
  - eine niedrigere Abstraktionsebene auswählen, um den Handlungsspielraum für die automatische Umplanung einzuschränken.

Auf der gewählten Ebene kann der Disponent erneut versuchen, die Aufgabe nach seinen Vorstellungen im Plan einzuordnen.

- Er kann Algorithmus 8 beenden, ohne eine Auswahl zu treffen und in weiteren Iterationen zunächst andere Aufgaben nach seinen eigenen Vorstellungen umplanen, um eine freie, d.h. konsistente Position für Aufgabe a zu schaffen.

3.7.2 Reduzierung des Planungsaufwandes durch den Einsatz der Werkzeuge

Die Anwendung des durch Algorithmus 5 und 8 definierten Lösungsschemas kann nach den in Abschnitt 3.2.2 beschriebenen Interaktionsmodellen erfolgen. Aus Algorithmus 8 und 9 können Verbesserungen für den wissensbasierten Planungsprozess bezüglich der in Abschnitt 3.2.3 identifizierten Defizite abgeleitet werden:

Aufwand der manuellen Planung: Mit Hilfe der Werkzeuge II und IV können Sackgassen und Backtracking-Vorgänge bei der manuellen Planung reduziert werden (Algorithmus 8, Zeile 12). Es wird verhindert, dass der Disponent Positionen auswählt, die zu keiner Lösung führen und es werden die Fälle reduziert, in denen der Disponent unwissentlich eine Position auswählt, die erst nach mehreren weiteren Zuweisungen einen Konflikt verursacht. Trotzdem kann das Auftreten von Sackgassen nicht ausgeschlossen werden. Die Konfliktauflösung wird jedoch durch die Werkzeuge III und V vereinfacht.

Qualität des Gesamtplans: Bei der Anpassung eines Plans kann der Disponent die Planungsfunktionen (Werkzeug III) einsetzen. Da sie eine Optimierung der Zielfunktionen anstreben, wird sichergestellt, dass die Anforderungen an die Qualität im ausreichenden Maß berücksichtigt werden. Je höher die gewählte Abstraktionsebene, desto mehr Spielraum zur Optimierung erhält der Computer.

Complacency-Effekt: Die Werkzeuge I bis V verringern den Aufwand der manuellen Planung. Die Visualisierung des Handlungsspielraums durch die Werkzeuge II und IV erleichtert zudem das Verständnis für die zugrunde liegenden Randbedingungen. Die Werkzeuge tragen somit dazu bei, dass der Disponent gut informierte Entscheidungen trifft und erleichtern es ihm, mehrere Sze-

narien durchzuspielen und Entscheidungsalternativen gegeneinander abzuwägen. Der Complacency-Effekt kann damit reduziert werden, denn für den Disponenten sinkt die Hürde, Planungsvorschläge des Computers zu modifizieren.

Planung durch Versuch und Irrtum: Eine Unterteilung in Abstraktionsebenen ermöglicht es dem Disponenten, Entscheidungen „so konkret wie nötig und so abstrakt wie möglich“ zu formulieren. Eine möglichst abstrakte Entscheidungsformulierung bewirkt, dass der Computer mehr Spielraum erhält, um eine Lösung zu finden. Eine Planung durch Versuch und Irrtum wird dadurch vermieden.

Kontrolle über das Ausmaß der Umplanung: Mit Hilfe von Werkzeug I, III und V kann der Disponent die automatische Umplanung für bestimmte Aufgaben auf bestimmte Planabschnitte ausgewählter Abstraktionsebenen begrenzen. Damit lassen sich unerwünschte Abweichungen vom Ausgangsplan vermeiden.

3.7.3 Abgrenzung zu den Richtlinien von Frayman (2001)

Bei der Gestaltung der Planungswerkzeuge und des neuen Lösungsschemas (Algorithmen 5 und 8) wurden die in Abschnitt 3.3.2 beschriebenen Interaktionsrichtlinien für Konfigurationsprobleme zum Vorbild genommen. Aufgrund der in Abschnitt 3.3.3 identifizierten Anforderungen wurden die Richtlinien 2, 3, und 6 an die Besonderheiten von Planungsproblemen angepasst:

Richtlinie 2: Um eine effiziente Auswahl von Teilbereichen aus den Wertebereichen von Variablen zu ermöglichen, wurde das Konzept einer Unterteilung in Planabschnitte auf mehreren Abstraktionsebenen entwickelt (Abschnitt 3.4.1). Alle Planungswerkzeuge berücksichtigen die Abschnittsunterteilung.

Richtlinie 3: Mit Werkzeug IV wurde eine Kategorisierung vorgeschlagen, die es dem Disponenten ermöglicht, das erforderliche Ausmaß der Umplanung zur Auflösung eines inkonsistenten Zustands abzuschätzen. Außerdem kann der Disponent die Umplanung mit den Werkzeugen I, III und V auf den gewünschten Abstraktionsebenen nach seinen eigenen Vorstellungen kontrollieren.

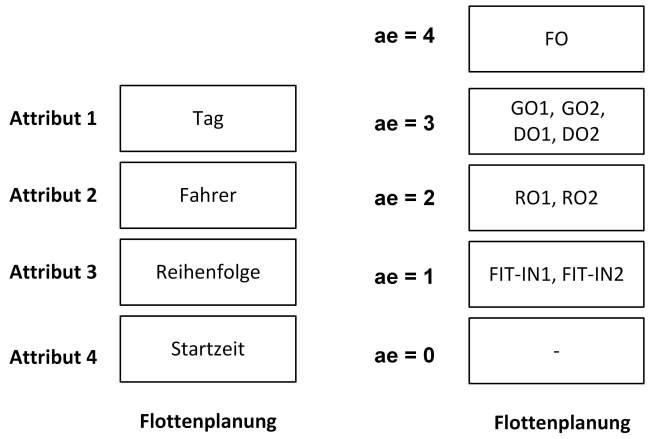


Abbildung 3.12: Zusammenfassung der Attribute und Abstraktionsebenen für die Flottenplanung (vgl. Beispiele 3.5 und 3.8)

Richtlinie 6: Die Backtracking-Freiheit wird durch Werkzeug II angenähert. Eine vollständige Backtracking-Freiheit kann aufgrund der Größe und Komplexität praktischer Planungsprobleme nicht garantiert werden, aber es wird eine Implementierung angestrebt, mit der innerhalb der zulässigen Antwortzeit so viele Inkonsistenzen wie möglich identifiziert werden können (vgl. Richlinie 7, Abschnitt 3.3.2).

3.7.4 Beispiel für die Anwendung von Algorithmus 8

Zur besseren Lesbarkeit von Beispiel 3.10 fasst Abbildung 3.12 nochmals die Abbildungen 3.8 und 3.10 zusammen.

Beispiel 3.10: Die Fallstudie der Beispiele 2.19 bis 2.21 lässt sich auf das durch die Algorithmen 5 und 8 definierte Lösungsschema abbilden, wenn ein VRPTW-Modell zugrunde gelegt wird (Abschnitt 2.2.3). Es wird die Anwendung der Werkzeuge II und IV auf den in Abbildung 3.12 gezeigten Abstraktionsebenen demonstriert.
Ausgangssituation ist ein vollständiger Plan. Es wird angenommen, dass die zu modifizierende Tour für den 01.08.2016 in der Arbeitszeit von 8:00 bis 16:30 Uhr geplant ist. Der Disponent versucht, einige Aufgaben einen Tag nach vorn zu verlegen (vgl. Beispiel 2.20), also auf den 31.07.2016. Es ergibt sich folgender Ablauf:

- Es gilt: $ae \leftarrow 2$ (Zeile 5), da eine Abschnittsunterteilung in Tage (horizontal) und Fahrer (vertikal) benötigt wird (vgl. Beispiel 3.8).
- Der Disponent wählt eine Aufgabe a_{select}, d.h. $v_{select} \leftarrow a_{select}$ (Zeile 6), aus der Liste *scheduled*. Das Zeitfenster von a_{select} sei $estart(a_{select})$=01.08.2016 10:00, $lstart(a_{select})$=04.08.2016 12:00 (vgl. Gleichung 2.14).
- Der Disponent lässt sich für Aufgabe a_{select} die Positionen im Plan kennzeichnen (Zeile 12, Werkzeug IV). Da er die Aufgabe für den gleichen Fahrer einen Tag nach vorn verlegen möchte, interessiert er sich für den Planabschnitt (31.07.2016, $a_{select_{res}}$). Auf Abstraktionsebene 2 ist dieser jedoch rot gekennzeichnet, da er aufgrund des Zeitfensters von a_{select} inkonsistent bezüglich der ursprünglichen Randbedingungen ist (Kategorie 4).
- Daher gilt: $positions = \emptyset$ und demzufolge auch $candidates = \emptyset$. Die Prozedur ist beendet. Es erfolgt der Übergang zu Algorithmus 5. Der Disponent stellt die ursprüngliche Startzeit wieder her (z.B. mit einer „Rückgängig"-Funktion).
- Er versucht in weiteren Iterationen ohne Erfolg, auf die gleiche Weise weitere Aufgaben nach vorn zu verlegen.

In Beispiel 2.21 wird ein anderer Auftrag a_{select} ausgewählt, der bei den Mitarbeitern X oder Y am gleichen Tag eingeplant werden soll. Die Mitarbeiter werden durch die Ressourcen res_x und res_y repräsentiert.

- $ae \leftarrow 0$ (Zeile 5), da der Disponent die konkrete Startzeit mitbestimmen möchte, um eine Einplanung am Nachmittag zu erreichen.
- $v_{select} \leftarrow a_{select}$ (Zeile 6)
- Der Disponent lässt sich für Aufgabe a_{select} die Positionen im Plan kennzeichnen (Zeile 12). Er betrachtet zunächst die Markierung der Positionen bei Ressource res_x. Auf Abstraktionsebene 0 sind dort alle Positionen mit Startzeiten am Nachmittag grau gekennzeichnet (Werkzeug IV, Kategorie 3), d.h. eine Umplanung anderer Aufgaben ist innerhalb des Abschnitts (01.08.2016, res_x) nicht möglich, um eine dieser Positionen konsistent zu machen.

- Der Disponent betrachtet nun die Markierung der Positionen bei Ressource res_y. Einige Positionen mit Startzeiten am Nachmittag sind dort (in grüner Farbe) als konsistent gekennzeichnet (Werkzeug II).
- Der Disponent weist nacheinander der Aufgabe a_{select} einige dieser Positionen probehalber zu und bewertet die resultierenden Touren (Zeilen 16 bis 18). Schließlich entscheidet er sich für eine Position und weist sie der Aufgabe abschließend zu (Zeilen 22 und 23).

Die Prozedur ist beendet. Es erfolgt der Übergang zu Algorithmus 5. Der Disponent setzt den Lösungsprozess mit einer anderen Aufgabe a fort.

Die Vorteile der Werkzeuge II und IV werden im obigen Beispiel besonders bei der Umsetzung von Beispiel 2.21 deutlich: Der Disponent erspart sich aufgrund der von Werkzeug IV vorgenommenen Kategorisierung aufwändige Umplanungsversuche bei Mitarbeiter X. Gleichzeitig werden freie Positionen bei Mitarbeiter Y durch Werkzeug II visualisiert.

3.8 Anwendung der Planungswerkzeuge auf höheren Aggregationsebenen

Die Werkzeuge I bis V können auf jeder Aggregationsebene des Planungsproblems angewendet werden. Auf Aggregationsebenen oberhalb der Feinplanungsebene bezieht sich der Einsatz der Werkzeuge nicht nur auf einzelne Aufgaben, sondern auf Aufgabengruppen (Aggregate). Die für Aufgabengruppen manuell oder automatisch getroffenen Planungsentscheidungen bilden Randbedingungen für die Aufgaben oder Aufgabengruppen niedrigerer Ebenen (vgl. Abschnitt 2.2). Im Gegensatz zur Feinplanung bleibt nach der Festlegung einer Startzeit für ein Aggregat in der Regel noch Handlungsspielraum für die Festlegung des Endzeitpunktes übrig. Die Planungswerkzeuge müssen daher sowohl die Ermittlung von Startzeiten, als auch die Ermittlung von Endzeiten unterstützen.
Aufgrund dieser Vorüberlegungen können die Werkzeuge auf höheren Aggregationsebenen wie folgt interpretiert werden:

Werkzeug I: Neben der Fixierung der Startzeit- und Ressourcenzuweisung sollte auch eine Fixierung des Endzeitpunktes ermöglicht werden. Außerdem sollte ein Aggregat auf einen bestimmten Zeitabschnitt fixiert werden können. In diesem Fall wird sowohl der frühestmögliche Beginn als auch das späteste mögliche Ende für die damit verknüpften Unteraggregate oder Aufgaben auf diesen Abschnitt beschränkt (vgl. Randbedingungen 2.4 und 2.5).

Werkzeuge III und V: Auch auf höheren Aggregationssebenen lassen sich Abstraktionsebenen und Planabschnitte bilden. Dementsprechend können auch die Werkzeuge III und V angewendet werden. Mit Hilfe von Werkzeug III kann z.B. innerhalb eines bestimmten Zeitabschnitts die Ermittlung von Start- und Endzeiten für ein oder mehrere Aggregate automatisch vorgenommen werden. Mit Hilfe von Werkzeug V können z.B. Aggregate zwischen bestimmten Zeitabschnitten umgeordnet werden.

Werkzeuge II und IV: Werkzeug II sollte je nach Bedarf des Disponenten eine Unterstützung bei der Wahl der Startzeit oder Endzeit anbieten. Wenn von Werkzeug II eine Start- oder Endzeit für ein Aggregat als inkonsistent oder konsistent gekennzeichnet wurde, bedeutet dies, dass die damit verbundenen Randbedingungen für die verknüpften Unteraggregate bzw. Aufgaben im aktuellen Feinplan nicht erfüllbar bzw. erfüllbar sind.
Wenn ein Aggregat einem bestimmten Planabschnitt zugeteilt wird, sollte automatisch überprüft werden, ob eine Umplanung der restlichen Aggregate dieses Abschnitts notwendig und möglich ist, sodass alle verknüpften Unteraggregate bzw. Aufgaben die aufgestellten Randbedingungen erfüllen können (Werkzeug IV).

Beispiel 3.11: Ein Betrieb stellt Produkte her, die sich aus mehreren Komponenten zusammensetzen (vgl. Beispiel 2.2). Für den Grobplan auf der Produktebene wurde der Planungshorizont eines Jahres in Quartalsabschnitte unterteilt. Der Disponent möchte ermitteln, ob es möglich ist, eine bestimmte Menge von Produkt X im zweiten Quartal herzustellen, ohne die bereits für dieses Quartal angeordneten Produktionsaufträge auf spätere Quartale verschieben zu müssen. Mit Werkzeug IV könnte sich folgende Kategorisierung ergeben:

Grün: Alle Aufträge können im vorgesehenen Produktionszeitraum (Startzeit - Endzeit) ausgeführt werden, eine Planänderung ist nicht notwendig.

Gelb: Der Produktionszeitraum muss für einige Aufträge verändert werden, die Produktion ist aber jeweils innerhalb des Quartals möglich.

Grau: Die Produktion muss für einige Aufträge auf ein anderes Quartal verschoben werden.

Rot: Die Einhaltung von Lieferterminen ist für andere Aufträge innerhalb des Planungshorizontes nicht möglich, wenn diese Planungsentscheidung getroffen wird.

3.9 Zusammenfassung: 9 Richtlinien für die Funktionsaufteilung in Planungssystemen

Im Folgenden werden alle Erkenntnisse, die über die ideale Entscheidungsunterstützung des Disponenten gewonnen wurden, in einem Satz von 9 Interaktionsrichtlinien zusammengefasst. Richtlinie R1 fordert die Ermittlung ein oder mehrerer Zuweisungsregeln (vgl. Abschnitt 3.6), damit die Planungswerkzeuge an die Besonderheiten des Anwendungsbereiches angepasst werden können. Die Richtlinien R2 bis R4 fassen die in Abschnitt 3.1.1 beschriebenen Mindestanforderungen an das Planungssystem zusammen. Die Richtlinien R5 bis R9 beschreiben die Planungswerkzeuge I bis V aus Abschnitt 3.4.

Richtlinie R1 (Ermittlung der Zuweisungsregeln):
Vor der Entwicklung eines Planungssystems sollten typische Zuweisungsregeln ermittelt werden, die zur wissensbasierten Lösung des Planungsproblems benötigt werden.

Richtlinie R2 (Manuelle Anpassung von Plänen):
Die Werte aller im Plan enthaltenen Entscheidungsvariablen sollten manuell anpassbar sein (vgl. Abschnitt 3.1.1).

Richtlinie R3 (Überprüfung der Einhaltung von Randbedingungen):
Das Planungssystem sollte manuelle Wertzuweisungen auf die Einhaltung von Randbedingungen überprüfen. Wenn die Verletzung einer Randbedingung festgestellt wird, sollte der Disponent darüber informiert werden, um welche Randbedingung es sich handelt und welche Variablen davon betroffen sind. Die Überprüfung und Information sollten nach jeder einzelnen Wertzuweisung stattfinden (vgl. Abschnitt 3.1.1).

Richtlinie R4 (Überprüfung der Einhaltung strukturierter Anforderungen):
Das Planungssystem sollte die Einhaltung strukturierter Anforderungen auf der Anforderungsebene der Abstraktionshierarchie (vgl. Abschnitt 2.4.2) überprüfen und den Disponenten nach jedem Zuweisungsschritt über deren Einhaltung oder Nicht-Einhaltung informieren. Außerdem sollte der Disponent nach jeder einzelnen Wertzuweisung über die Qualitätskennzahlen des (partiellen) Plans informiert werden (vgl. Abschnitt 3.1.1).

Richtlinie R5 (Fixierung, Werkzeug I):
Das Planungssystem sollte es ermöglichen, Randbedingungen für Entscheidungsvariablen festzulegen, die bei der automatischen Planung berücksichtigt werden müssen. Dieser Vorgang wird als **Fixierung** von Werten bezeichnet (vgl. Abschnitt 3.4.2).

R5.1: Alle Werte aus dem Wertebereich einer Entscheidungsvariablen können zur Fixierung ausgewählt werden. Die Fixierung ist für alle Variablen möglich. Das Planungssystem sollte darüber hinaus ggf. die Fixierung von Abschnitten des Wertebereichs ermöglichen.

R5.2: Fixierungen müssen bei der Verwendung einer Zuweisungsregel nach oben verkettet werden: Die Fixierung eines Wertes oder eines Abschnitts von Werten für Attribut j hat zur Folge, dass die Werte oder Abschnitte für die Attribute 1 bis $j-1$ für die zugehörige Aufgabe ebenfalls fixiert werden müssen (vgl. Abschnitt 3.6.2).

R5.3: Bei der anschließenden Anwendung der Interaktionsrichtlinien R2, R3, R4, R6, R7, R8 und R9 sollte das durch Fixierung erweiterte Planungsmodell zugrunde gelegt werden.

Richtlinie R6 (Unterstützung bei der Wertauswahl, Werkzeug II):
Das Planungssystem sollte den Disponenten während der Planung bei der Auswahl von Werten aus dem Wertebereich einer Entscheidungsvariable unterstützen (vgl. Abschnitt 3.4.3). Zu diesem Zweck sollte für eine vom Disponenten ausgewählte Aufgabe jede Position im Plan als „inkonsistent bezüglich der ursprünglichen Randbedingungen“, „inkonsistent bezüglich der bisherigen Zuweisungen“ oder „konsistent“ gekennzeichnet werden.

Richtlinie R7 (Abstraktionsebenen und Planungsfunktionen, Werkzeug III):
Das Planungssystem sollte eine Planung auf unterschiedlichen Abstraktionsebenen ermöglichen (vgl. Abschnitt 3.4.1). Auf jeder Ebene $ae, ae > 0$ sollte der Plan in eine Menge nicht-überlappender Abschnitte unterteilt werden, die jeweils eine bestimmte Menge zusammenhängender Positionen bzw. Abschnitte der nächst tieferen Ebene $ae - 1$ umfassen. Für jeden Abschnitt sollte eine Planungsfunktion bereitgestellt werden, die eine konfliktfreie, konsistente und lokal optimierte Anordnung der Aufgaben innerhalb des Abschnitts berechnet (vgl. Abschnitt 3.4.4). Die Anzahl der Ebenen und die Anordnung der Abschnitte kann aus einer Zuweisungsregel ermittelt werden (vgl. Abschnitt 3.6.3).

Richtlinie R8 (Erweiterte Unterstützung bei der Wertauswahl, Werkzeug IV):
Das Planungssystem sollte auf jeder Abstraktionsebene den Disponenten bei der Auswahl von Positionen (Ebene 0) oder Abschnitten (Ebene $i, i > 0$) für eine Aufgabe a unterstützen (vgl. Abschnitt 3.4.5). Zu diesem Zweck sollte jede Position bzw. jeder Abschnitt wie folgt gekennzeichnet werden:

- „konsistent“
- „inkonsistent bezüglich der bisherigen Zuweisungen, aber konsistent nach Anwendung der zugehörigen Planungsfunktion“
- „inkonsistent bezüglich der bisherigen Zuweisungen, aber konsistent nach Ausführung einer abschnittsübergreifenden Konfliktauflösung“ (manuell oder mit Werkzeug V)
- „inkonsistent bezüglich der ursprünglichen Randbedingungen“

Richtlinie R9 (Automatische Konfliktauflösung, Werkzeug V):
Das Planungssystem sollte auf jeder Abstraktionsebene eine automatische abschnittsübergreifende Auflösung von Konflikten ermöglichen (vgl. Abschnitt 3.4.6). Das Resultat sollte ein konfliktfreier und konsistenter (partieller) Plan sein, bei dem die ursprünglichen Positionen bzw. Abschnittszuweisungen der Aufgaben weitestgehend erhalten bleiben. Wenn die Beibehaltung der Position bzw. des Abschnitts für eine Aufgabe nicht möglich ist, sollte die Beibehaltung eines der übergeordneten Abschnitte der nächst höheren Abstraktionsebenen angestrebt werden.

Aufgrund der hohen Problemkomplexität können die Richtlinien R6 bis R9 ggf. nur unter Zeitbeschränkungen eingesetzt werden (vgl. Abschnitt 3.5). Die Unterstützung der Wertauswahl nach Richtlinie R6 basiert zudem in der Regel nur auf Verfahren zur Berechnung lokaler Konsistenz anstatt globaler Konsistenz. Es können daher u.U. nicht alle inkonsistenten Werte identifiziert werden.

3.10 Zusammenfassung

In diesem Kapitel wurde zur Beantwortung der Forschungsfrage ein Konzept zur Gestaltung der Funktionsaufteilung zwischen Mensch und Computer entwickelt, welches auf beliebige, der Modellstruktur gehorchende Planungsprobleme angewendet werden kann. Zunächst wurde unter Annahme einer herkömmlichen Funktionsaufteilung der Ablauf von Entscheidungsprozessen bei der Planung (Abschnitt 2.4.3) als integrierter Lösungsprozess formuliert, an dem Mensch und Computer beteiligt sind (Abschnitt 3.1.3). Dabei wurden Defizite im Interaktionsablauf erkennbar, die den Entwurf und die Bewertung von Alternativlösungen für den Disponenten zu einem zeitaufwändigen und fehleranfälligen Prozess werden lassen (Abschnitt 3.2).
Da der Interaktionsablauf bei der Planung Ähnlichkeiten zum Interaktionsablauf bei Konfigurationsproblemen aufweist (Abschnitt 3.3.1), können die Richtlinien zur Funktionsaufteilung bei Konfigurationsproblemen teilweise auf Planungsprobleme übertragen werden. Aufgrund der typischen Problemgröße und -komplexität sowie aufgrund von besonderen Anforderungen an die Entscheidungsunterstützung bei Planungsproblemen sind die Richtlinien jedoch nicht vollständig übertragbar und decken den Unterstützungsbedarf nicht vollständig ab (Abschnitt 3.3.3). Daher

wurde eine Spezialisierung und Ergänzung der Richtlinien für Planungsprobleme vorgenommen. Es wurden 5 Planungswerkzeuge konzipiert, die dem Disponenten bei der interaktiven Planung an der Benutzerschnittstelle zur Verfügung gestellt werden sollten (Abschnitt 3.4). Jedes Werkzeug beinhaltet eine bestimmte Form der Computerunterstützung zur Entscheidungsfindung oder Entscheidungsumsetzung an einem bestimmten Punkt im integrierten Lösungsprozess.
Die Defizite des herkömmlichen Lösungsprozesses sind mit der Integration der Werkzeuge konzeptionell beseitigt (Abschnitt 3.7). Die Werkzeuge fließen in einen Satz von Interaktionsrichtlinien ein (3.9). Sie beschreiben die Mindestanforderungen an die Gestaltung von interaktiver Optimierung für Planungssysteme und liefern damit eine Antwort auf die in Abschnitt 1.4.1 gestellte Forschungsfrage. Die Anwendbarkeit und Effektivität der Richtlinien wird in den folgenden Kapiteln anhand eines empirischen Anwendertests nachgewiesen.

Abgrenzung zu bestehender Funktionalität industrieller Planungssysteme

Die übliche Funktionalität industrieller Planungssysteme wurde in Abschnitt 1.3.3 beschrieben. Die Interaktionsrichtlinien R5 bis R7 lassen Rückschlüsse darüber zu, wie die bestehende Funktionalität angepasst werden sollte:

Fixierung (Richlinie R5): Die Fixierung entspricht der Funktionalität „manuelle Planungsentscheidungen“. Die formale Analyse des Entscheidungsprozesses in den Algorithmen 6 und 10 zeigt, worauf bei der Umsetzung der Fixierung stärker zu achten ist: Zum einen sind Fixierungen temporär, d.h. sie werden innerhalb eines Entscheidungsprozesses mehrmals gesetzt und geändert, um die gewünschten Alternativlösungen zu generieren. Das Setzen und Aufheben von Fixierungen sollte daher für alle Arten von Planungsentscheidungen ermöglicht und gleichzeitig einfach und intuitiv gestaltet werden, um die Erzeugung von Alternativlösungen zu beschleunigen. Zum anderen sollte in Abhängigkeit von den ermittelten Zuweisungsregeln die Fixierung von Zuweisungen zu Planabschnitten auf bestimmten Abstraktionsebenen ermöglicht werden.

Wertauswahlunterstützung (Richtlinie R6): Die Unterstützung der Wertauswahl lässt sich dem Konzept „Überwachung manueller Planungsentscheidungen“ zuordnen. Das bestehende Konzept zielt jedoch lediglich darauf ab, die Ver-

letzung von Randbedingungen im nächsten manuellen Planungsschritt zu verhindern. Richtlinie R7 sieht darüber hinaus vor, dass auch zukünftige Sackgassen, die erst nach mehreren weiteren Planungsschritten auftreten, im Voraus erkannt und vermieden werden (vgl. Abschnitt 3.4).

Planungsfunktionen (Richtlinie R7): Das Konzept der Planungswerkzeuge ähnelt dem Konzept der „partiellen Neuplanung“. Richtlinie R6 erweitert das Konzept, indem sie einen Ansatz zur Eingrenzung der Planabschnitte für die partielle Neuplanung vor dem Hintergrund der Entscheidungsunterstützung liefert. Zur Bestimmung der Planabschnitte werden Abstraktionsebenen genutzt, die aus einer Zuweisungsregel hergeleitet werden können (Abschnitt 3.6). Die partielle Neuplanung unterstützt damit die Erzeugung von Alternativlösungen mit einer beliebig abstrakten Sichtweise auf den Plan. Der Entscheidungsprozess wird effizienter, da der Disponent nicht relevante Details unterer Ebenen vernachlässigen kann und gleichzeitig der Handlungsspielraum des Computers zur Umsetzung der abstrakten Entscheidungen größer ist.

Die Richtlinien R2 bis R4 beschreiben grundlegende Funktionen, die bereits in den meisten Planungssystemen vorhanden sind. Die Richtlinien R1, R8 und R9 besitzen keine Korrespondenz zu bestehender Funktionalität.

4 Fallstudien zur Anwendung der Interaktionsrichtlinien

In diesem Kapitel wird der Einsatz der Interaktionsrichtlinien R1 bis R9 bei der Entwicklung von Planungssystemen demonstriert. Außerdem wird der Nachweis geführt, dass mit dem Einsatz des neu entwickelten Interaktionsmodells eine Verbesserung der Benutzererfahrung gegenüber den herkömmlichen Interaktionsmodellen (vgl. Abschnitt 3.2) erreicht werden kann. Zu diesem Zweck wurde anhand eines Prototypen zur Flottenplanung ein Anwendertest mit 80 Teilnehmern durchgeführt. Der Entwurf des Flottenplanungssystems und der Aufbau des Tests werden in Abschnitt 4.1 erläutert.
Um die Anwendbarkeit der Richtlinien auf weitere Planungsprobleme zu demonstrieren, werden in Abschnitt 4.2 mit Hilfe der Richtlinien Empfehlungen für die Weiterentwicklung eines Stundenplanungssystems (aus Stenzel (2012), Prototyp der BTU Cottbus-Senftenberg) entwickelt.

4.1 Fallstudie Flottenplanung: Prototyp und Anwendertest

In diesem Abschnitt wird ein praktischer Anwendungsfall der Flottenplanung betrachtet, der zur Durchführung des Anwendertests verwendet wird. Nach einer kurzen Einführung in die geltenden Planungsvorschriften und -ziele erfolgt die Beschreibung eines Prototyps, der von mir unter Anwendung der Interaktionsrichtlinien R1 bis R9 entwickelt wurde[1]. Anschließend werden in den Abschnitten 4.1.3

[1] Der Prototyp befindet sich derzeit unter der Adresse `http://comores.inf.hszg.de/eus_flottenplanung`, Stand: 17.07.2017. Die in Anhang A angegebene Adresse ist nicht mehr aktuell.

und 4.1.4 das Testziel und der Testaufbau erläutert. Die Testergebnisse werden in Abschnitt 4.1.5 vorgestellt. Die Details zur Vorbereitung und Durchführung des Tests sind im Testplan beschrieben, der in Anhang A beigefügt ist.

4.1.1 Einführung in den Anwendungsfall

Unternehmen und Organisationen, die alle oder einen Teil ihrer Dienstleistungen beim Kunden vor Ort erbringen, müssen den Einsatz ihrer Fahrzeuge nach wirtschaftlichen Gesichtspunkten koordinieren. Mit dieser Aufgabe werden Disponenten betraut, die im Rahmen der Flottenplanung Touren für jeden einzelnen Fahrer zusammenstellen (vgl. Beispiele 1.1, 1.2, 1.4, 2.11, 2.19 sowie Abbildung 1.2). Die vorliegende Fallstudie basiert auf dem in Abschnitt 2.2.3 beschriebenen VRPTW-Problem. Zusätzlich werden folgende Randbedingungen und Optimierungsziele zugrunde gelegt.

Randbedingungen:

1. Für jede Aufgabe ist ein Zeitfenster vorgegeben, welches innerhalb eines bestimmten Tages den frühest- und spätestmöglichen Startzeitpunkt festlegt.
2. Bestimmte Aufgaben dürfen nur von Fahrern (Ressourcen) mit einer bestimmten Qualifikation ausgeführt werden.
3. Bestimmte Aufgaben sind einem bestimmten Fahrer fest zugeordnet.
4. Innerhalb einer Tour müssen die Aufgaben überlappungsfrei angeordnet werden. Eine Aufgabe i kann erst ausgeführt werden, nachdem die vorhergehende Aufgabe $(i-1)$ abgeschlossen und die Fahrzeit zum Ort der Aufgabe i zurückgelegt wurde.
5. Die im Schichtplan festgelegte Arbeitszeit eines Fahrers darf nicht überschritten werden.
6. Die Tour eines Fahrers beginnt und endet am selben Tag an einem fest zugeordneten Anfangs- und Endort, der sich aus der Schicht des jeweiligen Fahrers ergibt.

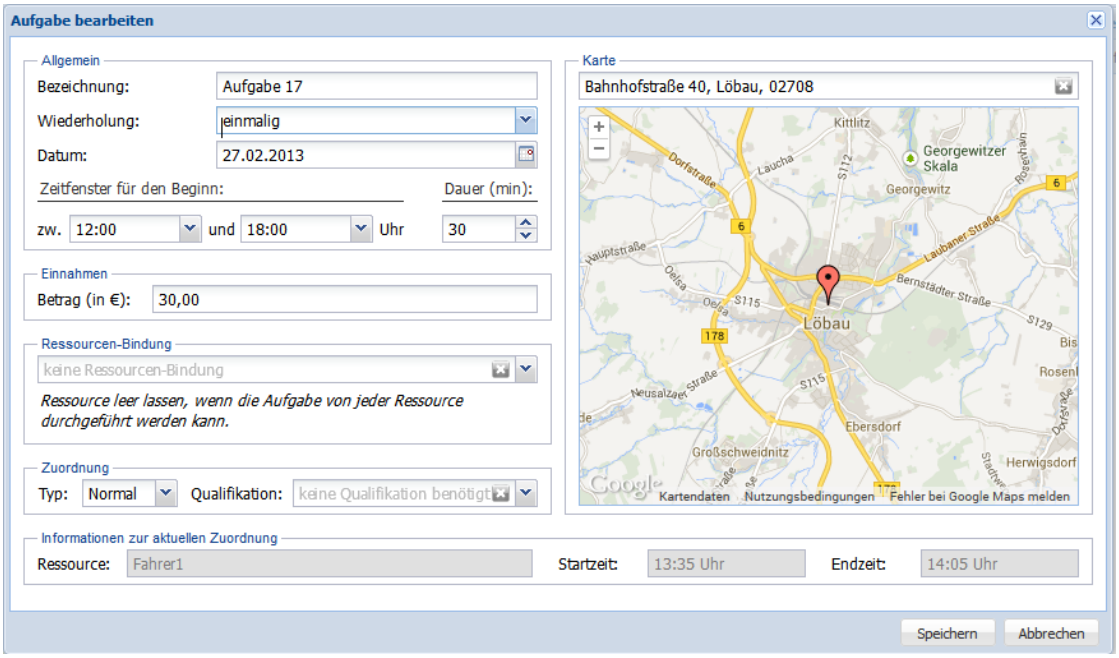


Abbildung 4.1: Dialog zum Anlegen einer neuen Aufgabe

Optimierungsziele:

Z1: Maximierung des Gewinns durch Minimierung der Fahrtkosten[2] für alle Touren. Der Gewinn ergibt sich aus der Summe der Einnahmen der ausgeführten Aufgaben abzüglich der Fahrtkosten für alle Touren.

Z2: Ausgleich der für die Ausführung der Touren benötigten Arbeitszeiten[3] für alle Fahrer.

Bei der Optimierung bleibt die vorgegebene Anzahl von Fahrern unverändert. Die Randbedingungen 1 bis 3 werden beim Anlegen der Aufgaben festgelegt. Dabei ist nur die Festlegung der Randbedingung 1 zwingend erforderlich. Abbildung 4.1 zeigt den Dialog zur Eingabe der Aufgabendaten. Die Eingabe der Arbeitszeiten und der Anfangs- und Endorte der jeweiligen Schichten erfolgt in einem separaten Schichtplanungsbereich des Planungssystems.
Für die Berechnung der Zielfunktionswerte für die Optimierungsziele Z1 und Z2 wird ein bestimmter Planungszeitraum (z.B. 4 Wochen) zugrunde gelegt, der im

[2]Es gilt: Fahrtkosten = Fahrtstrecke * Kilometerpauschale, die Kilometerpauschale ist im Planungssystem frei einstellbar.

[3]Benötigte Arbeitszeit = Zeitpunkt der Ankunft am Endort - Zeitpunkt der Abfahrt am Anfangsort

Prototyp frei eingestellt werden kann. Das Planungssystem stellt zwei Optimierungsoptionen bereit, die Z1 und Z2 in einer linearen Optimierungsfunktion unterschiedlich gewichten. Die Option „Fahrstrecke minimieren“ gewichtet Z1 und Z2 jeweils mit den Faktoren 1.0 und 0. Für die Option „Arbeitszeit ausgleichen“ werden die Faktoren 0.7 und 0.3 verwendet, da das Ziel der Gewinnmaximierung eine höhere Priorität besitzt.
Das Planungsmodell stellt verschiedene Aufgabentypen für die Berücksichtigung spezieller Planungsanforderungen bereit. Aufgaben vom Typ „Pause“ ermöglichen eine flexible, an der aktuellen Tour orientierte Einplanung von Arbeitspausen. Beim Anlegen einer Pause kann auf die Angabe eines Ortes verzichtet werden. In diesem Fall gilt der Ort der vorhergehenden Aufgabe in der Tour als Pausenort. Für besonders dringende oder kurzfristig einzuplanende Aufgaben kann der Typ „Ereignis“ festgelegt werden. Die Aufgabe erhält dadurch eine höhere Priorität und wird bei der automatischen Planung so zeitig wie möglich (unter Berücksichtigung des Zeitfensters) eingeplant.

4.1.2 Beschreibung der Benutzerschnittstelle

Einen Gesamtüberblick über die Benutzeroberfläche des Flottenplanungssystems bietet Abbildung 4.2. Die Oberfläche besitzt ein dreiteiliges Layout, welches aus den Abschnitten „Ressourcenplanung“, „verfügbare Aufgaben“ und „Karte“ zusammengesetzt ist. Im Abschnitt „Ressourcenplanung“ werden die Touren der Fahrer in einer horizontalen Zeitleiste dargestellt, die sich über alle Tage des eingestellten Planungshorizontes erstreckt. Graue Balken repräsentieren die Zeiträume, in denen normale Aufgaben ausgeführt werden. Für Pausen wird eine grüne und für Ereignisse eine rote Einfärbung des gesamten Balkens verwendet. Die Fahrzeiten zwischen jeweils zwei Aufgaben werden als braune, schmalere Balken dargestellt.
Der ein- und ausklappbare Abschnitt „verfügbare Aufgaben“ dient als Ablage für noch unverplante Aufgaben. Alle Aufgaben können per Drag-and-Drop vom Plan in die Ablage verschoben werden und umgekehrt. Der Abschnitt „Karte“ stellt eine geografische Ansicht der Touren bereit. Wenn sich der Planungshorizont über mehr als einen Tag erstreckt, besteht die Möglichkeit, die Anzeige der Routen nach einem bestimmten Tag zu filtern.
Die nach Richtlinie R1 zugrunde gelegte Zuweisungsregel wurde in Beispiel 3.8 eingeführt (siehe auch Abbildung 3.8). Im Folgenden werden die nach den Richtlinien

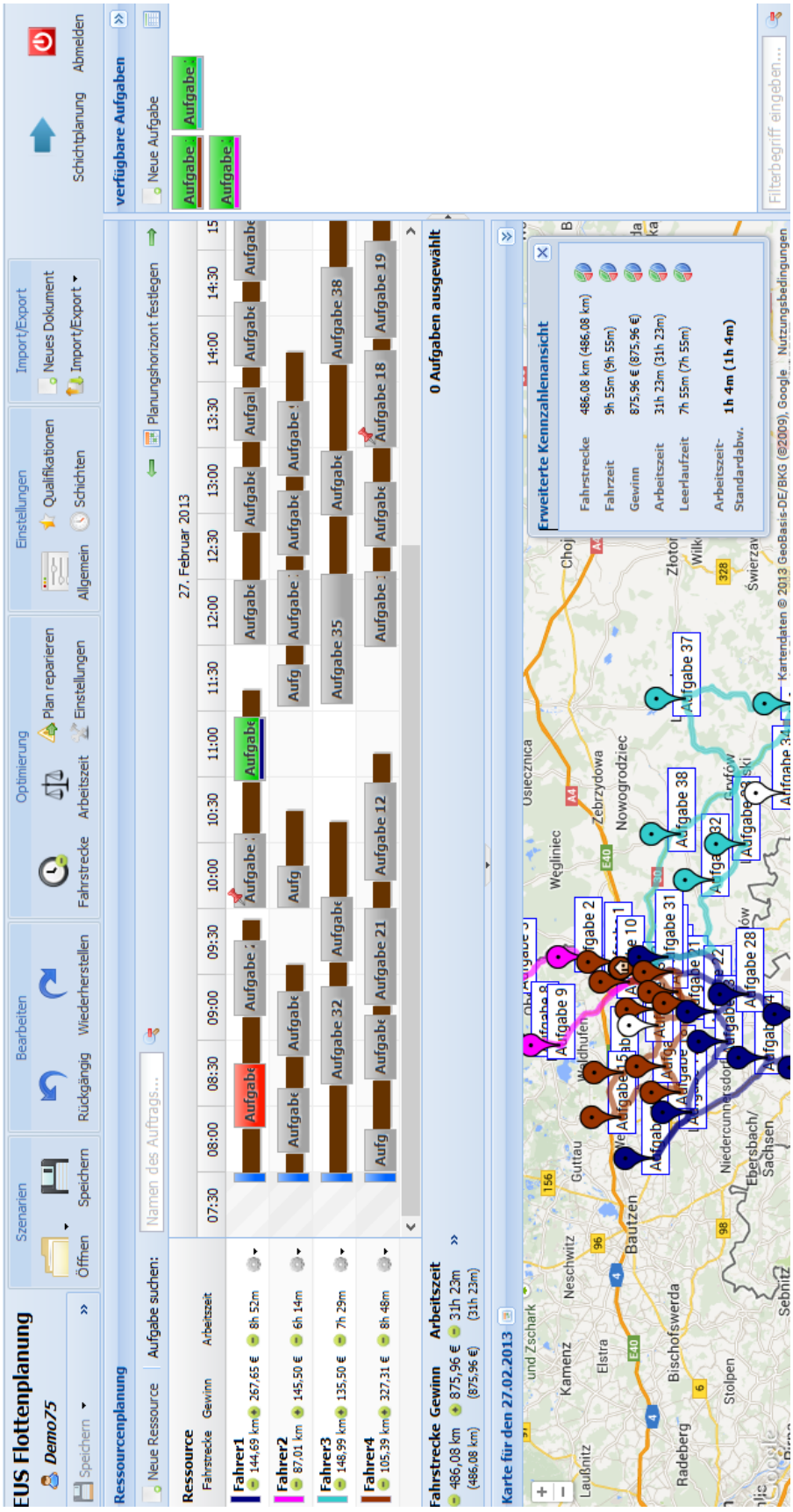


Abbildung 4.2: Benutzerschnittstelle für die Flottenplanung

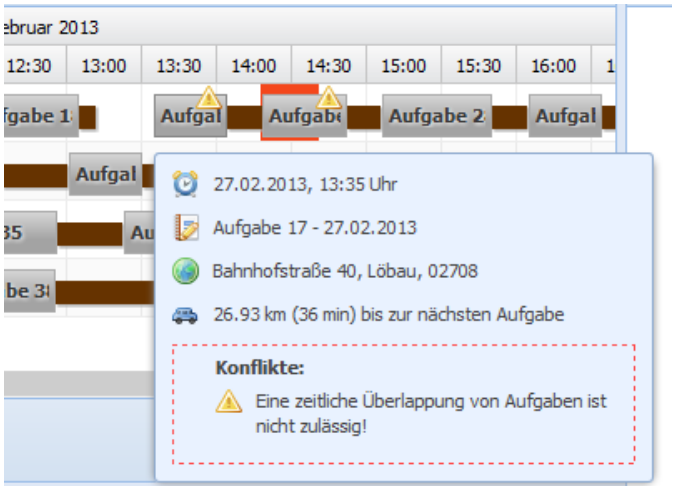


Abbildung 4.3: Fehlermeldung bei einer zeitlichen Überlappung von zwei Aufgaben, die durch das Überfahren eines Warndreiecks mit der Maus aufgerufen werden kann (die rechte Aufgabe liegt im rot markierten Bereich über dem braunen Fahrzeitbalken der linken Aufgabe).

gestalteten Interaktionsmöglichkeiten beschrieben.

Manuelle Anpassung von Plänen nach Richtlinie R2

Eine manuelle Änderung von Planungsentscheidungen kann per Drag-and-Drop im Plan vorgenommen werden. In einem einzigen Drag-and-Drop-Vorgang können dabei Tag, Ressource, Position und Startzeit einer Aufgabe verändert werden.

Überprüfung der Einhaltung von Randbedingungen nach Richtlinie R3

Auf die Verletzung von Randbedingungen wird durch ein gelbes Warndreieck hingewiesen, welches auf dem Balken der betroffenen Aufgaben angebracht wird. Wird ein solches Dreieck mit der Maus überfahren, erscheint eine Fehlermeldung, aus der entnommen werden kann, um welche Randbedingung es sich handelt. Bei einer Überlappung von zwei Aufgaben wird zusätzlich der Zeitraum, an dem beide Aufgaben zur gleichen Zeit stattfinden, rot hervorgehoben (Abbildung 4.3).

Überprüfung der Einhaltung strukturierter Anforderungen nach Richtlinie R4

Die Kennzahlen Fahrzeit, Gewinn und Arbeitszeit werden im Abschnitt „Ressourcenplanung“ als Summe der Werte, die sich aus den einzelnen Touren ergeben, dargestellt. Für eine bessere Nachvollziehbarkeit werden die Werte der einzelnen Touren ebenfalls aufgeführt. Zusätzlich steht die „erweiterte Kennzahlenansicht“ als frei verschiebbares Popup-Fenster zur Verfügung. Sie enthält weitere Kennzahlen, wie z.B. Fahrstrecke, Leerlaufzeit und Arbeitszeit-Standardabweichung. Letzere ist ein Maß dafür, wie ausgeglichen die Arbeitszeiten der Fahrer sind (Z2).
Die Kennzahlen werden automatisch nach jeder automatisch oder manuell vorgenommenen Planänderung aktualisiert. Darüber hinaus werden sie auch während eines Drag-and-Drop-Vorgangs aktualisiert, sobald die Position oder die Ressourcenzuweisung der bewegten Aufgabe geändert wird. Dabei wird das Ziel verfolgt, eine möglichst flüssige Planung zu ermöglichen und eine häufige Unterbrechung des Drag-and-Drop-Vorgangs zu vermeiden. Das Verhalten des Systems kann in Abbildung 4.8 nachvollzogen werden: Aufgabe 28 wird während des Drag-Vorgangs hinter Aufgabe 38 verschoben, wobei sich die Gesamtfahrstrecke von 488,78 km auf 514,84 km erhöht. Gleichzeitig wird die Länge des Fahrzeitbalkens zwischen Aufgabe 38 und Aufgabe 28 entsprechend angepasst. Diese Hilfestellung erleichtert es dem Disponenten, eine Überlappung von Aufgaben zu vermeiden.

Explizite Fixierung nach Richtlinie R5

Das Planungssystem ermöglicht eine Fixierung für Attribut 2 (Ressource bzw. Fahrer) und 4 (Startzeit) der Zuweisungsregel[4] (vgl. Abbildung 3.8). Die Fixierung der Ressource auf Ebene 2 ist nur im Eigenschaftsdialog einer Aufgabe möglich (Abbildung 4.1). Aufgaben, die fest an einen Fahrer gebunden sind, werden im Plan an der unteren Seite des Balkens durch einen Streifen in der Farbe des Fahrers gekennzeichnet (Abbildung 4.2).
Die Fixierung von Ressource und Startzeit auf Ebene 4 kann im Plan über das Kontextmenü einer Aufgabe ausgeführt werden. Sie wird durch eine Stecknadel, die am Aufgabenbalken angebracht wird, grafisch visualisiert (z.B. Aufgabe 18 in Abbildung 4.2). Es besteht die Möglichkeit, eine Gruppe markierter Aufgaben auf einmal zu fixieren.

[4]Eine Fixierung der Reihenfolge ist nicht möglich.

Planung und Konfliktauflösung nach Richtlinie R7

Im Planungssystem wurden mit Ausnahme von „Day Optimization 1“ (DO1), „Day Optimization 2“ (DO2) und „Resource Optimization 2“ (RO2) alle Planungsfunktionen aus Abbildung 3.10 (Beispiel 3.8) umgesetzt. Ein Aufruf von „Full Optimization“ (FO) für den gewählten Planungshorizont ist im Menü im Abschnitt „Optimierung“ über die Buttons „Fahrzeit“ (Zielfunktion Z1) und „Arbeitszeit“ (Zielfunktion Z2) möglich. Um eine Gruppenoptimierung „Group Optimization 1“ (GO1) oder „Group Optimization 2“ (GO2) durchzuführen, muss zunächst eine Gruppe von Aufgaben im Plan und/oder in der Ablage markiert werden. Dies geschieht mit gedrückter STRG-Taste oder über ein Auswahlrechteck, welches über eine Gruppe von Aufgaben aufgezogen werden kann. Der Aufruf der Optimierung kann dabei auf dem gleichen Weg wie der Aufruf der vollständigen Optimierung (FO) im Menü erfolgen. Auf diese Weise wird der Erwartung der Anwender entsprochen, dass sich eine bestimmte Aktion auf eine Teilmenge von Objekten beschränkt, wenn diese durch Markierung hervorgehoben wurden. Sie resultiert aus dem gewohnten Umgang mit Betriebssystem-Programmen, wie z.B. dem Windows Explorer. Alternativ ist der Aufruf auch über das Kontextmenü einer beliebigen markierten Aufgabe möglich (Abbildung 4.4).
Direkt nach dem Aufruf der Gruppenoptimierung wird der Nutzer in einem Popup-Dialog dazu aufgefordert, zwischen GO1 und GO2 zu wählen (Abbildung 4.5). Die Option „Reihenfolge beibehalten“ (GO2) ist die flexiblere Variante, bei der die zeitlichen Abstände zwischen den bereits eingeplanten Aufgaben bei Bedarf vergrößert werden können, um eine Einplanung der markierten Aufgaben zu ermöglichen.
Der Aufruf der Ressourcenoptimierung (RO1) ist im Abschnitt „Ressourcenplanung“ direkt neben den Kennzahlen der einzelnen Fahrer möglich (Abbildung 4.2 links in der Mitte). Ein Klick auf einen grünen Button $\oplus$ oder $\ominus$ bewirkt die Optimierung der jeweiligen Tour nach der entsprechenden Kennzahl.
Die Planungsfunktion FIT-IN1 dient zur Beseitigung von Planungsfehlern innerhalb einer Ressource. Sie wird daher als Option „Konflikt auflösen“ im Kontextmenü oder im Menü der Ressource angeboten, wenn ein Zeitfenster- oder Überlappungskonflikt vorliegt (Abbildung 4.6 und Abbildung 4.7). Wenn die vorgegebene Reihenfolge nicht beibehalten werden kann, werden minimale Abweichungen von der Reihenfolge zugelassen (Richtlinie R8, vgl. Abschnitt 3.6). Die Funktion FIT-IN2 wird unter dem Namen „Plan reparieren“ im Menü bereitgestellt (Abbildung 4.2 oben im Menüabschnitt „Optimierung“).

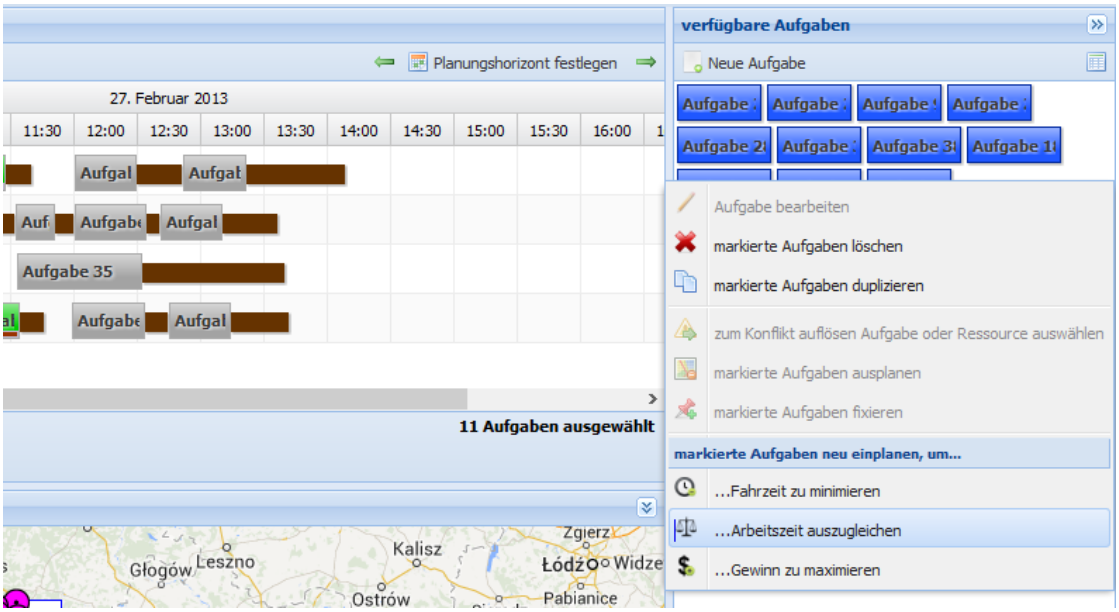


Abbildung 4.4: Markierung von Aufgaben (blau) und Aufruf der der Gruppenoptimierung nach der Zielfunktion Z2 („Arbeitszeit ausgleichen“).

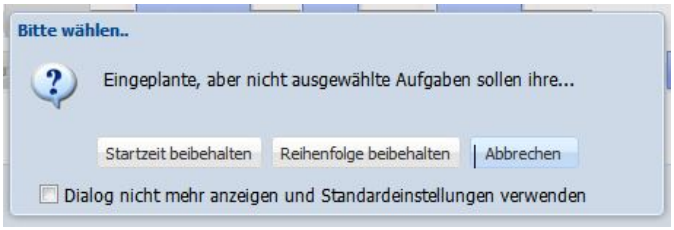


Abbildung 4.5: Popup-Dialog zur Auswahl von GO1 („Startzeit beibehalten“) oder GO2 („Reihenfolge beibehalten“).

Explizit fixierte Zuweisungen bleiben bei allen Optimierungsfunktionen erhalten. Wenn das Planungssystem beim Aufruf einer beliebigen Planungsfunktion keine Lösung findet, wird der Zustand des Plans vor dem Aufruf der Funktion wiederhergestellt. Der Anwender wird in einer Meldung darauf hingewiesen, dass keine Lösung gefunden werden konnte[5].

[5]Im Planungssystem ist zudem die Einstellung eines Timeouts für die Optimierung möglich. Wenn dieser vom Optimierungsalgorithmus überschritten wird, erhält der Anwender eine Mitteilung, dass in der vorgegebenen Zeit keine Lösung gefunden werden konnte.

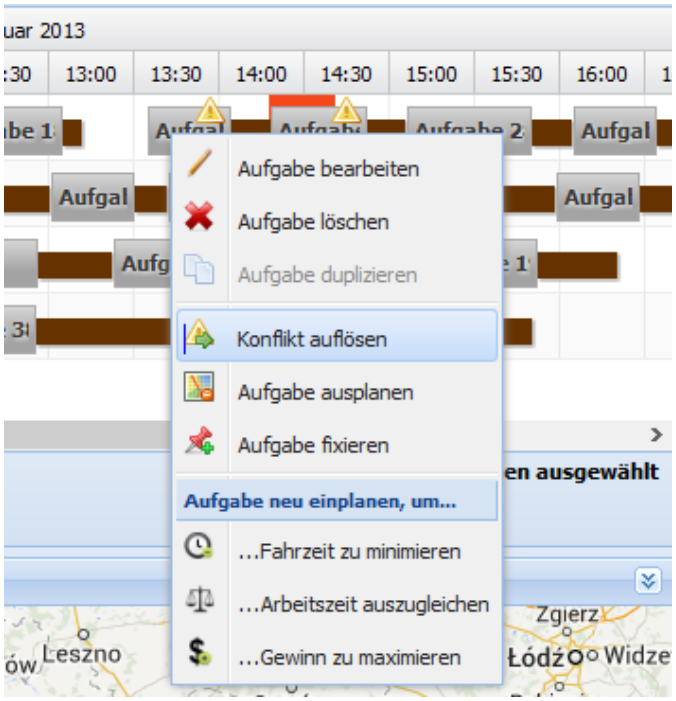


Abbildung 4.6: Planungsfunktion FIT-IN1 zur Auflösung von Überlappungsfehlern (Aufruf über das Kontextmenü)

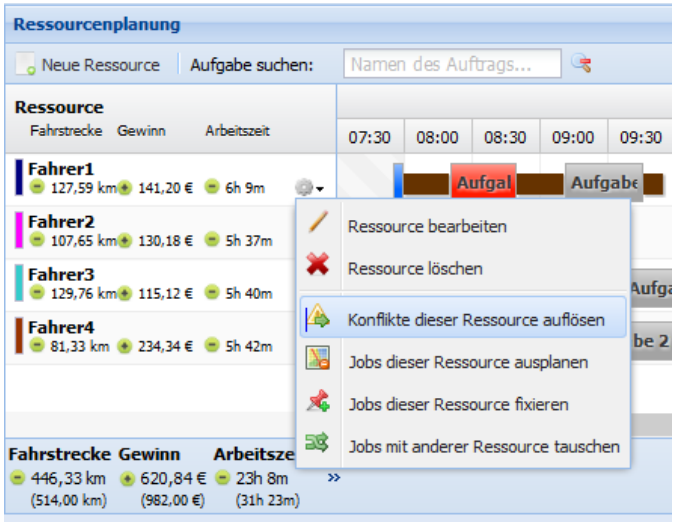


Abbildung 4.7: Planungsfunktion FIT-IN1 zur Auflösung von Überlappungsfehlern (Aufruf über das Ressourcenmenü)

Unterstützung bei der Wertauswahl nach R6

Die Zuweisung einer Startzeit für eine Aufgabe per Drag-and-Drop erfüllt R6, indem Zeiträume, die außerhalb des Zeitfensters der Aufgabe liegen, rot hinterlegt werden (Abbildung 4.8). Zeiträume, die eine überlappungsfreie Einplanung in die Tour ermöglichen, werden nach R6 grün hinterlegt. Eine gelbe Hinterlegung kennzeichnet inkonsistente Zeiträume, die im aktuellen Zustand des Plans nicht als Startzeit für die betrachtete Aufgabe verwendet werden können. Nach Richtlinie R6 werden zudem Zeitleisten von Ressourcen, die nicht über die erforderliche Qualifikation verfügen oder die nicht der fest zugewiesenen Ressource entsprechen, rot hinterlegt.

4.1.3 Testziel

In Abschnitt 3.2 wurden Defizite herkömmlicher Interaktionsmodelle identifiziert, die den Aufwand der wissensbasierten Planung erhöhen. Die Defizite beeinträchtigen die Planung in verschiedenen Situationen: vor der Planung, nach der Planung oder während der Planausführung (vgl. Abschnitt 2.4.3). Die in Abschnitt 3.4 konzipierten Interaktionswerkzeuge zielen darauf ab, die Defizite zu beseitigen und den Aufwand der wissensbasierten Planung in jeder Situation zu minimieren. Der hier beschriebene Anwendertest verfolgt zwei wesentliche Ziele: Zum einen soll nachgewiesen werden, dass die für jede Planungssituation hergeleiteten Defizite in praktischen Planungsszenarien tatsächlich beobachtet werden können. Zum anderen soll die Reduzierung des Planungsaufwands beim Einsatz der Interaktionswerkzeuge messbar nachvollzogen werden. Die Testhypothese, die aufgestellt wird, lautet daher: *Der Einsatz der Interaktionswerkzeuge führt zu einer Reduzierung des Aufwands der wissensbasierten Planung im Vergleich zu herkömmlichen Interaktionsmodellen.*

Einschränkungen

Als Testteilnehmer wurden Studenten aus verschiedenen Fakultäten der Hochschule Zittau/Görlitz eingesetzt. Da sie kein echtes Expertenwissen über die Disposition besaßen, musste die Intelligenzphase der Entscheidungsprozesse übersprungen

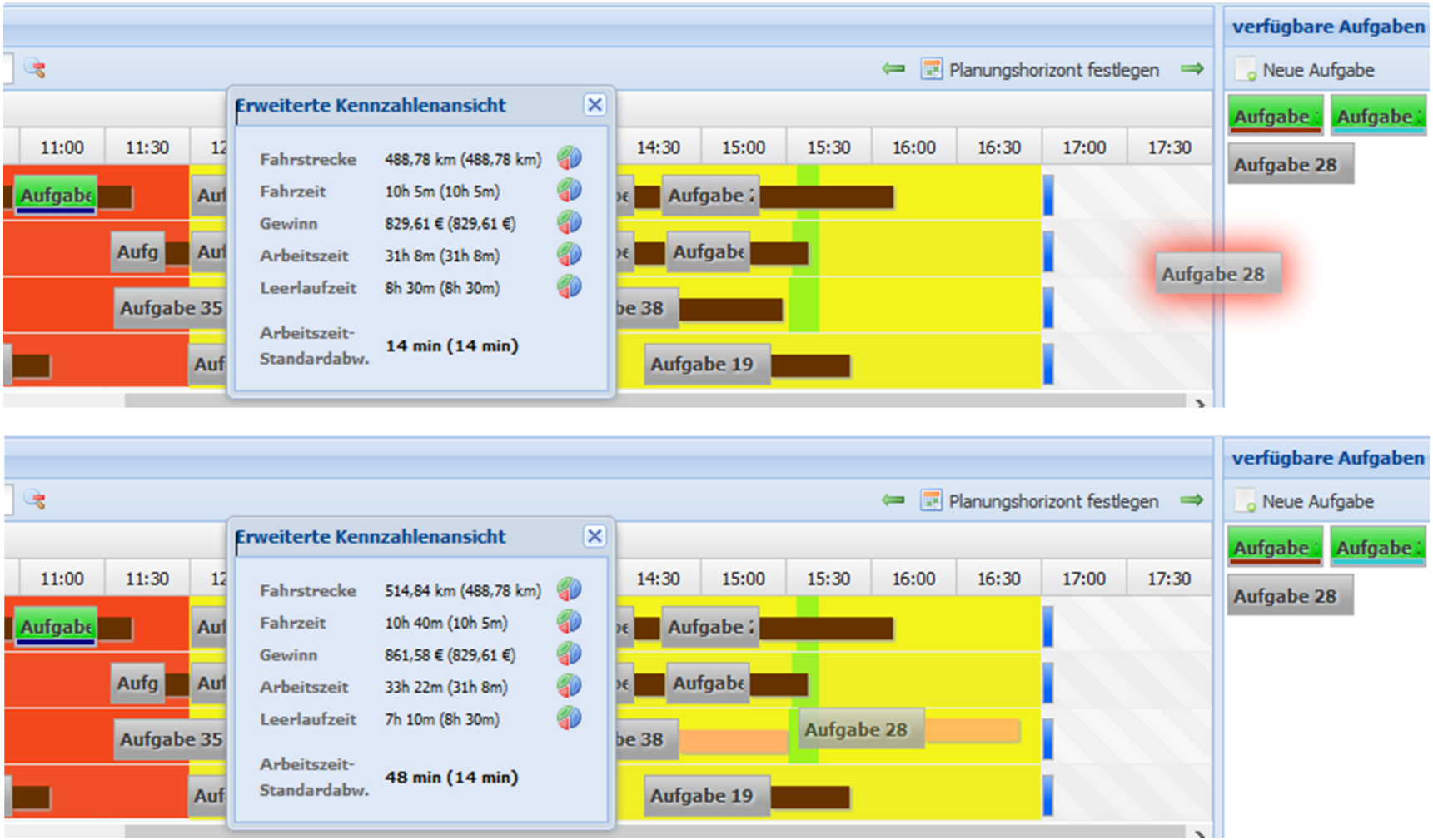


Abbildung 4.8: Farbliche Hinterlegung des Plans nach R9 sowie Aktualisierung der Fahrzeitbalken und Kennzahlen während eines Drag-and-Drop-Vorgangs für Aufgabe 28. Obere Abbildung: Übergang von der Ablage in den Plan. Untere Abbildung: Ankopplung an Aufgabe 38.

werden (vgl. Abschnitt 2.4.3). Stattdessen wurden die Planungsentscheidungen, die zur Überführung der unstrukturierten Anforderungen in strukturierte Anforderungen erforderlich sind, in der Aufgabenstellung fest vorgegeben. In der Entwurfsphase des Entscheidungsprozesses wurden die Alternativlösungen hinsichtlich der Zielfunktion Z1 (Fahrtkosten) verglichen.
Werkzeug IV (Richtlinie R8) wurde nicht in den Anwendertest integriert, da die Abschnittszuteilungen der Aufgaben in den Aufgabenstellungen vorgegeben war und somit keine Entscheidungsunterstützung benötigt wurde. Werkzeug V (Richtlinie R9) wurde ebenfalls nicht in den Anwendertest integriert, da eine Umplanung mit minimaler Abweichung von der Ausgangslösung nicht Bestandteil der Testaufgaben war. Außerdem besaßen die Studenten nicht genügend Expertenwissen, um einzuschätzen, ob die vom System vorgeschlagenen Planabweichungen tolerierbar sind.

Metriken zur Messung des Aufwands

Um den Aufwand zu messen, wurden bei der Lösung von Planungsproblemen durch die Testteilnehmer verschiedene Metriken erfasst:

- Die Metrik *Erfolgsrate* gibt an, welcher Anteil der Teilnehmer ein Planungsproblem mit einem bestimmten Interaktionsmodell erfolgreich lösen konnte. Eine Lösung ist erfolgreich, wenn eine vorgegebene Spezifikation unstrukturierter Anforderungen vollständig im Plan umgesetzt ist.

- Die Metrik *Zeit* gibt an, wie viele Minuten ein Teilnehmer zur erfolgreichen Lösung eines Planungsproblems benötigt.

- Eine erfolgreiche Umsetzung unstrukturierter Anforderungen ist nur dann zufriedenstellend, wenn die Qualität des Gesamtplans nicht beeinträchtigt wird. Die Metrik *Qualität* wurde daher als Maß für die Akzeptanz der erzeugten Pläne einbezogen.

Die Metriken Erfolgsrate und Zeit haben sich bei der quantitativen Bewertung der Benutzererfahrung bewährt (Tullis und Albert, 2008). Ein hoher Aufwand ist also durch eine niedrige Erfolgsrate, eine hohe Bearbeitungszeit und durch eine niedrige Qualität der erzeugten Pläne gekennzeichnet. Durch eine Videobeobachtung der Teilnehmer wurde der Aufwand darüber hinaus auch qualitativ gemessen.

4.1.4 Testaufbau

Der Anwendertest wurde in zwei Durchgängen durchgeführt, die im November 2011 und im Dezember 2012 stattfanden. Nach dem ersten Testdurchgang wurden Mängel in der Gebrauchstauglichkeit des Planungssystems beseitigt, d.h. die grafische Darstellung und die Bedienbarkeit der Interaktionselemente wurde verbessert[6]. Die im ersten und zweiten Durchgang eingesetzten Interaktionsmodelle sind in Abbildung 4.9 dargestellt.
In Testdurchgang 1 (35 Teilnehmer) stand die manuelle Planung im Vordergrund. In der Systemkonfiguration „None“ wird die Wertauswahl nicht durch den Computer unterstützt, während die Konfigurationen M1 und M2 jeweils eine Computerunterstützung nach Werkzeug II bzw. R6[7] bereitstellen. Die restlichen Konfigurationen sind den Interaktionsmodellen wie folgt zugeordnet (vgl. Abschnitt 3.2.2):

- M3 → Automatische Planung mit manueller Nachbearbeitung
- M4 → Automatische Planung unter Berücksichtigung zusätzlicher Randbedingungen (umgesetzt mit Fixierung nach Werkzeug I bzw. R5)
- M5 → Automatische Planung unter Berücksichtigung zusätzlicher Randbedingungen (umgesetzt mit den Werkzeugen I, II und III bzw. R5, R6 und R7)

Die Konfigurationen M1 bis M4 können den herkömmlichen Interaktionsmodellen zugeordnet werden. Das neu entwickelte, an eine Zuweisungsregel gekoppelte Interaktionsmodell ist durch M5 vertreten. Anhand der Modelle „None“ bis M2 soll zusätzlich untersucht werden, inwieweit die Richtlinie R6 dazu geeignet ist, den Aufwand der manuellen Planung zu reduzieren.
Der Testaufbau wurde in beiden Durchgängen so gestaltet, dass die Teilnehmer

[6] Dazu gehört u.a. ein leichteres Auswählen mehrerer Aufgaben für die Optimierung und eine bessere Menüstruktur, die das Auffinden von Planungsfunktionen erleichtert.

[7] Bei diesem Test wurde R6 in einer eingeschränkten und in einer vollständigen Variante umgesetzt. Die eingeschränkte Variante 1 für M1 hebt nur den ursprünglichen Wertebereich einer Variablen hervor (im Anhang A unter dem Begriff „Constraint-Hervorhebung“), die vollständige Variante 2 für M2 zeigt dem Nutzer den durch Propagierung der bisherigen Zuweisungen reduzierten Wertebereich (im Anhang A unter dem Begriff „Erweiterte Constraint-Hervorhebung (ECH)“).

Testdurchgang 1

	None	M1	M2	M3	M4	M5
Constraint-Hervorhebung		●	●	●	●	●
Erweiterte Constraint-Hervorhebung			●	●	●	●
Fixierung					●	●
FIT-IN, RO, GO						●
FO				●	●	

Testdurchgang 2

	M1	M2	M3	M4	M5
Erweiterte Constraint-Hervorhebung	●	●	●	●	●
Fixierung			●	●	●
FIT-IN, RO, GO				●	●
FO		●	●		●

Abbildung 4.9: Konfiguration des Flottenplanungssystems in Testdurchgang 1 und 2 durch Aktivierung/Deaktivierung von Funktionen[8]

die ihnen zugewiesenen Tests nacheinander mit den Konfigurationen M1 bis M5 lösten (vgl. Anhang A). Die schrittweise Hinzunahme von Interaktionselementen sollte die Einarbeitung der Teilnehmer erleichtern. Sie wurden dazu ermutigt, in jedem Test zusätzliche Interaktionselemente einzusetzen.
In Testdurchgang 2 (45 Teilnehmer) lag der Schwerpunkt auf der automatischen Planung. Das Interaktionsmodell „manuelle Planung“ ist unter Bereitstellung von R6 als Konfiguration M1 vertreten. Die Konfigurationen M2 bis M4 entsprechen den Konfigurationen M3 bis M5 im ersten Durchgang. Konfiguration M5 ergänzt M4 durch Hinzunahme der Planungsfunktion FO. Die vollautomatische Planung wurde erst im letzten Test freigeschaltet, nachdem sich die Teilnehmer bereits mit den restlichen Planungsfunktionen vertraut machen konnten. Auf diese Weise sollte ein einseitiger Einsatz von FO verhindert werden, der aus mangelndem Verständnis der restlichen Funktionen resultiert. Stattdessen sollten die Teilnehmer auf der Grundlage ihrer bisherigen Erfahrungen frei entscheiden können, gemäß Konfiguration M3 oder M4 zu planen.
Funktionen, die in einer bestimmten Konfiguration nicht zur Verfügung standen, wurden im Planungssystem deaktiviert (ausgegraut).

[8]Die rechte Tabelle entspricht Bild 1 im Testplan (Anhang A). Die Funktionen FIT-IN, RO und GO entsprechen dabei den Features „Konfliktauflösen“, „Ressourcenplanung“ und „Gruppenplanung“. Die Funktionen waren in der Ausprägung FIT-IN1, FIT-IN2, RO1, GO1 und GO2 verfügbar (vgl. Abschnitt 4.1.2).

Durchgang	Titel	Kategorie	Ebenen
1	Test 1	Neuplanung	Reihenfolge
1	Test 2	Neuplanung	Ressource, Startzeit
1	Test 3	Umplanung	Startzeit
1	Test 4	Umplanung	Ressource, Reihenfolge
1	Test 5	Umplanung	Startzeit
1	Test 6	Neuplanung	Reihenfolge
2	Test 1	Umplanung	Startzeit
2	Test 2	Umplanung	Ressource
2	Test 3	Neuplanung	Reihenfolge
2	Test 4	Umplanung	Ressource
2	Test 5	Neuplanung	Ressource
2	Test 6	Neuplanung	Reihenfolge, Startzeit

Tabelle 4.1: Merkmale der Tests in Durchgang 1 und 2

Testaufgaben

Für jeden Testdurchgang wurden 6 verschiedene Tests bzw. Planungsprobleme entworfen. Jeder Test fordert die Teilnehmer dazu auf, ein Flottenplanungsproblem für einen Arbeitstag, 4 Fahrer und 20 bis 30 Aufgaben an verschiedenen geografischen Standorten zu lösen. Die vollständigen Aufgabenstellungen des zweiten Testdurchgangs sind im Testplan in Anhang A enthalten. Ein Überblick über die Aufgabenstellungen des ersten Durchgangs ist bei Prenzel und Ringwelski (2012a) zu finden.

Die Aufgabenstellungen lassen sich in die Kategorien „Umplanung“ und „Neuplanung“ einteilen. Bei Tests zur Neuplanung befindet sich ein Großteil der Aufgaben zunächst in der Ablage und muss vom Teilnehmer eingeplant werden. Bei Tests zur Umplanung wird dem Teilnehmer ein vollständiger oder nahezu vollständiger Plan präsentiert, der modifiziert werden muss. In beiden Kategorien muss jeweils eine Reihe von Anforderungen auf verschiedenen Abstraktionsebenen berücksichtigt werden, die aus dem Expertenwissen eines Disponenten resultieren könnten. Tabelle 4.1 gibt für jeden Test die relevanten Ebenen an (vgl. Beispiel 3.8).

Die eingesetzten Planungsprobleme besitzen (aufgrund der Zeitfenster der Aufgaben) einen hohen Schwierigkeitsgrad, d.h. sie sind nicht global konsistent und weisen eine niedrige Lösungsdichte auf. Damit wird sichergestellt, dass die manu-

elle Lösung der Testaufgaben nicht trivial ist.
Die Testaufgaben spiegeln typische, in Abschnitt 2.4.1 besprochene Szenarien zur Anwendung von menschlichem Expertenwissen im Kontext der Arbeitsumgebung wider (Abschnitt 2.4.2). Da das zugrunde liegende Flottenplanungsproblem zudem der in Abschnitt 2.2 beschriebenen Modellstruktur gehorcht (vgl. Abschnitt 2.2.3 - VRPTW), können die Testergebnisse auf alle Planungsprobleme mit dieser Modellstruktur übertragen werden.

4.1.5 Auswertung und Diskussion der Testergebnisse

Die Testteilnehmer wurden zu 5 Peergruppen der Größe 7 (Durchgang 1) bzw. 9 (Durchgang 2) zusammengestellt. Pro Test wurde jeder Konfiguration mindestens eine Peergruppe zugeordnet (vgl. Anhang A sowie Prenzel und Ringwelski (2012a)). Gleichzeitig führte jede Peergruppe alle 6 Tests in unterschiedlichen Konfigurationen durch (vgl. Anhang A). Für jede Konfiguration wurden die Metriken Erfolgsrate, Zeit und Qualität erfasst.
Die Testteilnehmer wurden dazu aufgefordert, die Umplanung bzw. Neuplanung für jeden Test gemäß Aufgabenstellung durchzuführen. Ein Test galt als erfolgreich abgeschlossen, wenn die Aufgabenstellung erfüllt war und der erzeugte Plan keine Konflikte enthielt. Pläne mit verletzten Randbedingungen wurden von der Testauswertung (hinsichtlich Erfolgsrate, Qualität und Zeit) ausgeschlossen.
Die Interaktionen der Teilnehmer wurden als Video aufgezeichnet (vgl. Anhang A). Aus den Aufzeichnungen konnten Erkenntnisse über die Vorgehensweise der Teilnehmer zur Lösung der Tests gewonnen werden. Die Ergebnisse beider Testdurchgänge werden im Folgenden präsentiert und interpretiert.

Erfolgsrate

Der durchschnittliche Anteil der Testpersonen einer Peergruppe, die einen Test erfolgreich abgeschlossen hat, ist ein Maß für den Aufwand der Planung mit einer bestimmten Systemkonfiguration. Die Erfolgsrate wurde für jede Systemkonfiguration über alle Gruppen gemittelt.
Abbildung 4.10 zeigt die Verteilung der durchschnittlichen Erfolgsrate über alle 6 Systemkonfigurationen. In allen Konfigurationen, in denen die automatische

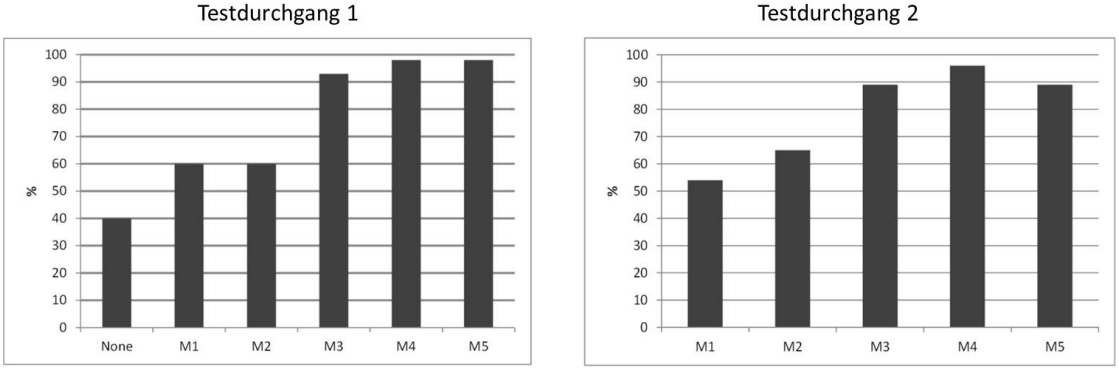


Abbildung 4.10: Durchschnittliche Erfolgsrate der Testteilnehmer pro Peergruppe

Planung nicht zur Verfügung steht, beträgt die Erfolgsrate in beiden Durchgängen nicht mehr als 60%. Konfigurationen, in denen automatisch erzeugte Pläne manuell nachbearbeitet werden müssen (M3 in Durchgang 1, M2 in Durchgang 2), weisen ebenfalls eine geringere Erfolgsrate auf. Der Unterschied zwischen Testdurchgang 1 (ca. 90%) und Testdurchgang 2 (65%) ist auf die einzelnen Aufgabenstellungen und die in den Tests verwendeten Planungsdaten (z.B. Zeitfenster der Aufgaben) zurückzuführen.

Die Ergebnisse bestätigen die Annahme, dass die manuelle Lösung von Planungsproblemen eine für Menschen ungeeignete Aufgabe ist (vgl. Abschnitt 3.2.1 und Abschnitt 3.2.3). Eine Entscheidungsunterstützung nach Richtlinie R6 (M1, M2 in Durchgang 1 und M1 in Durchgang 2) trägt nur geringfügig zur Erhöhung der Erfolgsrate bei. Die Gründe für das Scheitern der Planung waren bei der Beobachtung der Teilnehmer deutlich erkennbar: In Konfiguration „None“ wurde ein Großteil der Drag-and-Drop-Vorgänge fehlerhaft abgeschlossen, da den Teilnehmern weder das Zeitfenster der Aufgabe noch der erforderliche Abstand zu den bereits eingeplanten Aufgaben aus dem Kopf bekannt war[9]. Um einen gültigen (partiellen) Plan zu erstellen, mussten die Aufgaben häufig mehrmals im Plan verschoben werden.

Die Hervorhebung von Einfügepositionen mit Werkzeug III nach R6 (vollständige Variante „Enhanced Constraint Highlighting“) ermöglichte es den Teilnehmern, sehr schnell gültige partielle Pläne zu erzeugen. Sie freuten sich, wenn beim Drag-and-Drop ein oder mehrere grüne „Lücken“ erschienen, die ihnen signalisierten, dass eine weitere Aufgabe aus der Ablage „untergebracht“ werden konnte. Das

[9]Die dynamische Aktualisierung der Fahrzeitbalken wurde in dieser Konfiguration ebenfalls deaktiviert.

Vertrauen in die Vorschläge des Computers führte jedoch dazu, dass geografische Gesichtspunkte bei der Planung z.T. völlig vernachlässigt wurden. Teilweise wurde die Karte sogar ausgeblendet. Es wurden partielle Pläne mit einer sehr schlechten Qualität erzeugt, indem z.B. geografisch weit auseinander liegende Aufgaben demselben Fahrer zugeordnet wurden. Ab einem bestimmten Punkt konnten dadurch keine weiteren Aufgaben mehr innerhalb ihrer Zeitfenster dem Plan hinzugefügt werden, die Teilnehmer landeten also in einer Sackgasse. Sie waren häufig nicht mehr dazu motiviert, die Planung von vorn zu beginnen und brachen den Test an dieser Stelle ab.
Mit Hilfe des Modells „Automatische Planung unter Berücksichtigung zusätzlicher Randbedingungen“ konnten dagegen nahezu alle Teilnehmer ihre Tests erfolgreich abschließen (M4 in Durchgang 1, M3 in Durchgang 2). Die in Abschnitt 3.2.3 für dieses Modell identifizierten Defizite konnten nur teilweise beobachtet werden. Die Ursache ist in der Spezifikation der Testaufgaben zu suchen: Für die studentischen Teilnehmer wurden die Anforderungen so genau spezifiziert, dass sie leicht mittels Fixierung umgesetzt werden konnten. Im realen Umfeld ist jedoch zu erwarten, dass Anforderungen nur vage formuliert vorliegen und die konkrete Umsetzung erst in der Entwurfsphase des Entscheidungsprozesses erprobt werden muss (vgl. Abschnitt 2.4.3). Dabei können die Defizite einer Planung nach „Versuch und Irrtum“ auftreten.
Beim Einsatz des neu entwickelten Modells (M5 in Durchgang 1, M4 und M5 in Durchgang 2) lag die Erfolgsquote ebenfalls über 90 Prozent. Dies ist ein Nachweis für die Effektivität des Modells. Sein Einsatz verringert den Aufwand im Vergleich zu den herkömmlichen Interaktionsmodellen M1 bis M3. In Testdurchgang 2 kann ein leichter Rückgang der Erfolgsquote bei M5 beobachtet werden. Vermutlich wollten einige Teilnehmer die Tests mit der Planungsfunktion FO auf Knopfdruck lösen und stießen dabei auf die gleichen Probleme, die auch bei M2 beobachtet werden konnten (vgl. Abschnitt 3.2.3).

Qualität der Pläne - erzielte Gesamtfahrzeit

Die Testteilnehmer wurden dazu aufgefordert, die Gesamtfahrstrecke, d.h. die Summe der Fahrzeiten der 4 zu planenden Touren, unter Berücksichtigung der temporären Randbedingungen zu minimieren. Die Systemkonfigurationen wurden

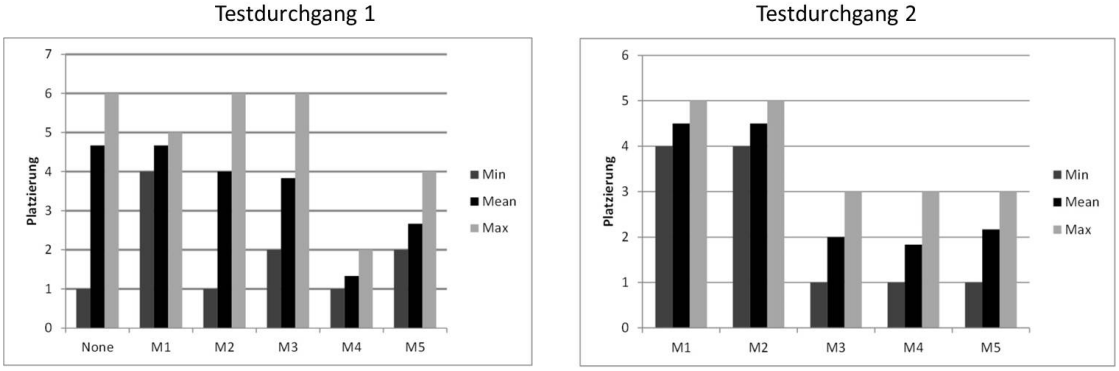


Abbildung 4.11: Durchschnittliche (Mean), beste (Min) und schlechteste (Max) Platzierung der Systemkonfigurationen in allen 6 Tests, gemessen an der erzielten Gesamtfahrzeit der Pläne.

für jeden Test nach der mit ihnen im Durchschnitt erzielten Qualität geordnet[10]. Platz 1 wurde für die Konfiguration vergeben, mit der im Schnitt die beste Qualität erreicht wurde. Abbildung 4.11 zeigt, welche Platzierung jede Konfiguration bei allen 6 Tests im Durchschnitt einnimmt. Zusätzlich wird jeweils die beste und die schlechteste Platzierung angegeben.
Mit den Konfigurationen, die Fixierung ermöglichen (M4 und M5 in Durchgang 1, M3 bis M5 in Durchgang 2), wurde die beste Qualität erzielt. Die schlechtere Qualität der manuell bearbeiteten Pläne lässt sich mit der hohen Schwierigkeit der Planungsprobleme begründen. Die Teilnehmer versuchten häufig, vor der Planung mit Hilfe der Karte günstige Touren mit geringer Fahrstrecke zu ermitteln. Ihre Umsetzung scheiterte jedoch häufig an den Zeitfenstern der Aufgaben, die es verhinderten, die vorgesehene Reihenfolge beizubehalten. Bei dem Versuch, alle Randbedingungen zu erfüllen, war es den Teilnehmern nahezu unmöglich, weiterhin auf die Qualität zu achten.
Neben der Karte wurde auch die Kennzahlenansicht für die Minimierung der Fahrstrecke genutzt. Bei der Einplanung von Aufgaben aus der Ablage achteten die Teilnehmer darauf, eine Position zu wählen, bei der sich die Kennzahl „Fahrstrecke“ nur geringfügig erhöht. Die bestmögliche Qualität des Gesamtplans wurde mit dieser Strategie häufig nicht erreicht. Die Teilnehmer waren nicht in der Lage, die Auswirkungen ihrer lokal günstigen Planungsentscheidungen auf die globale Qualität abzuschätzen.

[10]Die durchschnittlich erzielten Fahrstrecken können für jeden Test und jede Konfiguration in Anhang B eingesehen werden.

Die Ergebnisse zeigen, dass eine Fixierung in Kombination mit FO (M4 in Durchgang 1, M3 in Durchgang 2) und die Planungsfunktionen (Werkzeug III, M5 in Durchgang 1, M4 und M5 in Durchgang 2) bei der Umsetzung konkret formulierter Anforderungen gleichermaßen zur Optimierung geeignet sind (vgl. Auswertung zur Metrik „Erfolgsrate“).

Benötigte Zeit zur Lösung der Tests

Die benötigte Zeit zur Bearbeitung eines Tests in einer bestimmten Systemkonfiguration wurde über alle Teilnehmer eines Durchgangs gemittelt. Die Ergebnisse sind für alle Konfigurationen in Abbildung 4.12 dargestellt. Es ist erkennbar, dass die Bearbeitungszeit sinkt, sobald die automatische Planung zur Verfügung steht und die Interaktionsrichtlinien angewendet werden (M4 und M5 in Durchgang 1, M3 bis M5 in Durchgang 2). Das Ergebnis kann auf einen geringeren Aufwand bei der Planung zurückgeführt werden, die aus dem Einsatz der Richtlinien resultieren.
In Testdurchgang 2 kann ein Anstieg der Bearbeitungszeit bei M5 beobachtet werden. Dies ist vergleichbar mit dem Abfall der Erfolgsrate, der ebenfalls bei M5 beobachtet wurde. Auch hier gilt die Vermutung, dass einige Teilnehmer beim Einsatz von FO mit Problemen bei der nachträglichen Manipulation einer automatisch generierten Lösung zu kämpfen hatten (vgl. Auswertung zur Metrik „Erfolgsrate“ sowie Abschnitt 3.2.3).
Die vom Computer benötigte Zeit zur automatischen Lösung der Planungsprobleme war vernachlässigbar kurz. Es wurde der bei Prenzel und Ringwelski (2011) beschriebene Algorithmus eingesetzt.

Auswertung von Konfiguration M5 (Durchgang 2)

Konfiguration M5 nimmt eine Sonderstellung ein, da sie den Teilnehmern den freien Zugriff auf alle Interaktionselemente ermöglicht. Den Teilnehmern war es freigestellt, in dieser Konfiguration die Funktionen FIT-IN1, FIT-IN2, RO1, GO1 und GO2 zu verwenden oder ausschließlich Fixierung in Kombination mit FO einzusetzen. Abbildung 4.13 zeigt die durchschnittliche Häufigkeit der Verwendung der entsprechenden Interaktionselemente. Es ist erkennbar, dass der Einsatz der

[11] Alle Konfidenzintervalle wurden mit einem Konfidenzniveau von 90% berechnet.

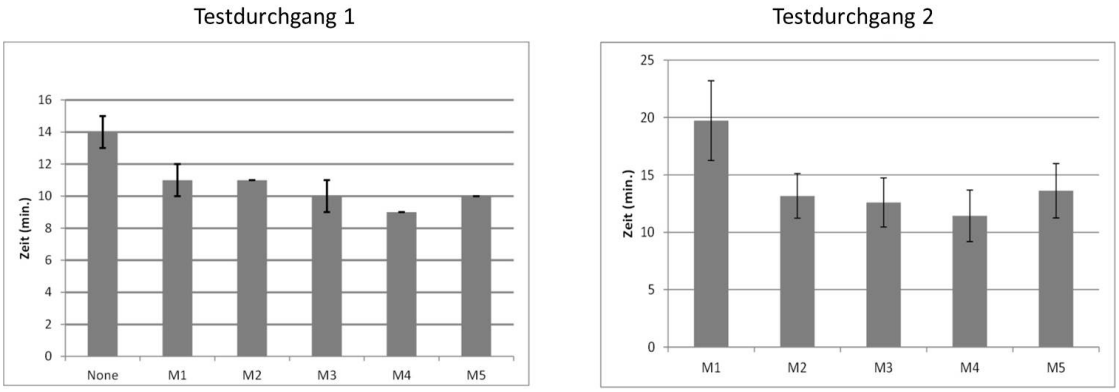


Abbildung 4.12: Durchschnittlich von einem Teilnehmer benötigte Zeit zur Lösung eines Tests (mit Konfidenzintervallen[11])

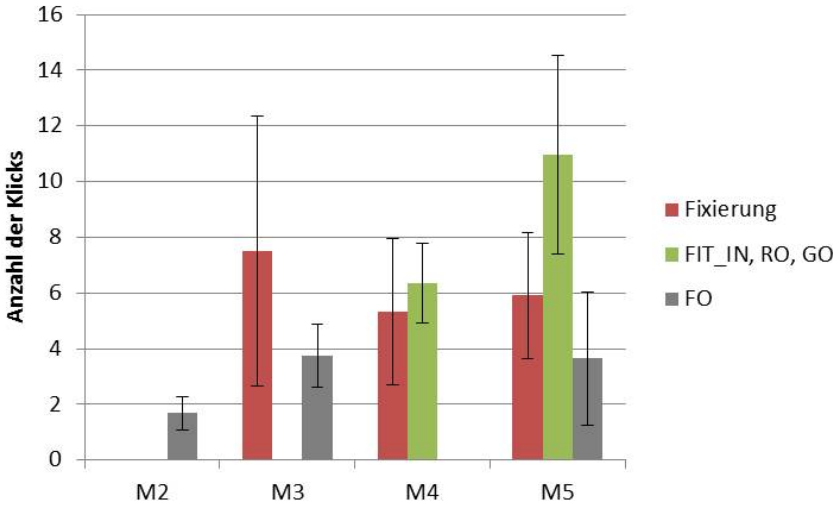


Abbildung 4.13: Durchschnittliche Häufigkeit der Verwendung von Fixierung und Planungsfunktionen pro Testteilnehmer (Testdurchgang 2)

Funktionen auf den Zuweisungsebenen 2 bis 4 in Konfiguration M5 im Vergleich zu Konfiguration M4 zunimmt. Dies lässt sich damit erklären, dass die Teilnehmer nach dem erstmaligen Einsatz in M4 mit den Funktionen besser vertraut sind und sie besser einzusetzen wissen. Außerdem zeigt die Häufigkeit der Verwendung, dass die Funktionen die gewünschte Benutzererfahrung bieten und nicht umgangen werden. Gleichzeitig werden sowohl die Fixierung als auch die vollautomatische Planung weiterhin in Konfiguration M5 angewendet.

4.1.6 Fazit

Die Ergebnisse des Anwendertests zeigen am Beispiel der Flottenplanung, dass es Anwendern große Schwierigkeiten bereitet, mit den herkömmlichen Interaktionsmodellen wissensbasierte Planungsprobleme zu lösen. Die Modelle bieten keine Unterstützung, um bei der Umsetzung zusätzlicher Randbedingungen den Aufwand zu reduzieren, d.h. Sackgassen vorauszusehen oder die Qualität des Gesamtplans zu optimieren. Mit Hilfe des neu entwickelten Interaktionsmodells, insbesondere durch den Einsatz von Fixierung und Planungsfunktionen auf der Basis einer Zuweisungsregel, können diese Defizite ausgeglichen werden. Dies wurde anhand einer höheren Erfolgsrate, einer kürzeren Bearbeitungszeit und einer besseren Qualität der Pläne nachgewiesen. Die Ergebnisse können auf alle Planungsprobleme übertragen werden, die auf die in Abschnitt 2.2 beschriebene Modellstruktur abgebildet werden können, da für sie die Interaktionsrichtlinien in der gleichen Weise angewendet werden können.

4.2 Fallstudie Universitätsstundenplanung: Empfehlungen zur Weiterentwicklung eines bestehenden Systems

In diesem Abschnitt wird ein prototypisches Stundenplanungssystem betrachtet, welches am Lehrstuhl für Programmiersprachen und Compilerbau der Brandenburgischen Technischen Universität Cottbus - Senftenberg im Rahmen einer Studien- und einer Bachelorarbeit entwickelt wurde (Höckner, 2011; Stenzel, 2012). Ziel ist es, die Stundenplanung für die Fakultät MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik mit ca. 15 Bachelor- und Masterstudiengängen durch Computerunterstützung zu erleichtern. Sie wird bisher von einem Mitarbeiter (im Folgenden als „Planer“ bezeichnet) mit einem Zeitaufwand von mehreren Wochen manuell durchgeführt. Zunächst wird der Ablauf der interaktiven Planung an der Benutzeroberfläche des entwickelten Systems beschrieben. Anschließend werden Empfehlungen zur Weiterentwicklung des Systems unter Berücksichtigung der Interaktionsrichtlinien gegeben.

4.2.1 Ablauf der Stundenplanung mit dem bestehenden System

Das Stundenplanungssystem ermöglicht sowohl eine automatische Planung als auch eine manuelle Bearbeitung von Plänen. Im Folgenden werden beide Funktionen kurz vorgestellt.

Automatische Planerstellung

Das Planungssystem ermöglicht eine automatische Stundenplanung für ein Winter- oder Sommersemester. Das verwendete Planungsmodell ist bei Höckner (2011) zu finden. Die Planung läuft in zwei Schritten ab. Zunächst werden die Vorlesungen eingeplant, wobei Randbedingungen berücksichtigt werden, die einen reibungslosen Ablauf der Lehre absichern (z.B.: *Für Studenten einer Fachrichtung gleichen Semesters dürfen keine zwei obligatorischen Kurse zur gleichen Zeit stattfinden.*) bzw. der Einhaltung von Studienordnungen dienen (z.B.: *Der Stundenplan muss Studenten aus früheren Jahrgängen eine Wiederholung von Kursen bei nachfolgenden Jahrgängen gestatten*). Nach der Einplanung werden die Vorlesungstermine von der Software fixiert, sodass sie im nächsten Planungsschritt nicht mehr verändert werden können. Im zweiten Planungsschritt werden Übungen, Praktika und Seminare eingeplant.
Die Software erzeugt stets einen gültigen Plan. Dieser ist für den Anwender nachvollziehbar, da der zweistufige Planungsprozess der typischen Vorgehensweise eines menschlichen Planers nachempfunden ist.
Bei Bedarf kann für die automatische Planung der abgespeicherte Plan des Vorjahres oder ein Teil dieses Plans zugrunde gelegt werden. Um den von den Studenten und Dozenten gewohnten Unterrichtsablauf beizubehalten, wird bei der automatischen Planung versucht, eine Umplanung der im Vorjahresplan enthaltenen Lehrveranstaltungen weitestgehend zu vermeiden.
Ein Ausschnitt aus einem vollständigen Stundenplan ist in Abbildung 4.14 dargestellt. Die Ansicht kann u.a. nach bestimmten Lehrveranstaltungen, Räumen oder Mitarbeitern gefiltert werden. Um einen neuen Plan zu erstellen, muss der Planer zunächst die für das jeweilige Semester erforderlichen Lehrveranstaltungen auswählen (Button „Veranstaltungen wählen“). Außerdem kann der Planer Räume und Dozenten für die Einplanung in bestimmten Blöcken sperren. Dies erfolgt über den Menüeintrag „Constraints“, über den ein entsprechender Dialog

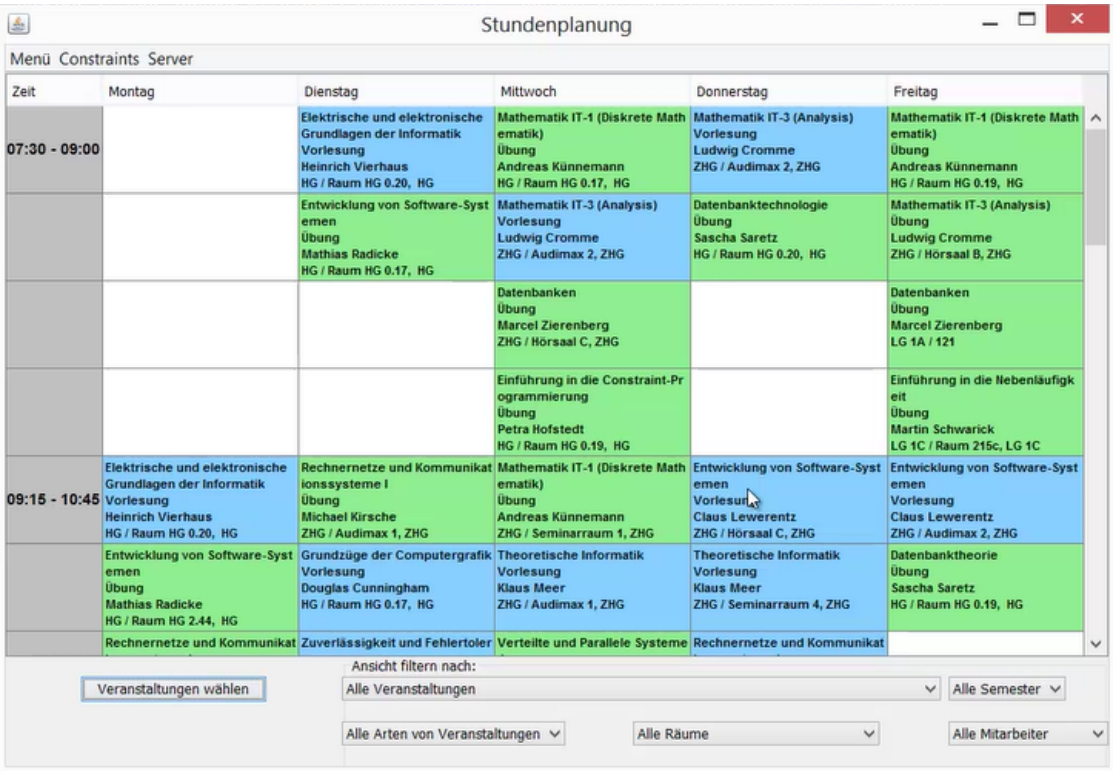


Abbildung 4.14: Plantafel der Stundenplanung für alle Lehrveranstaltungen des Studiengangs Informatik (Ansicht für den Planer)

Abbildung 4.15: Verbot (rot) und Freigabe (grün) für die Verwendung von Räumen in bestimmten Blöcken. Der Dialog kann analog für Dozenten aufgerufen werden.

aufgerufen werden kann (Abbildung 4.15).

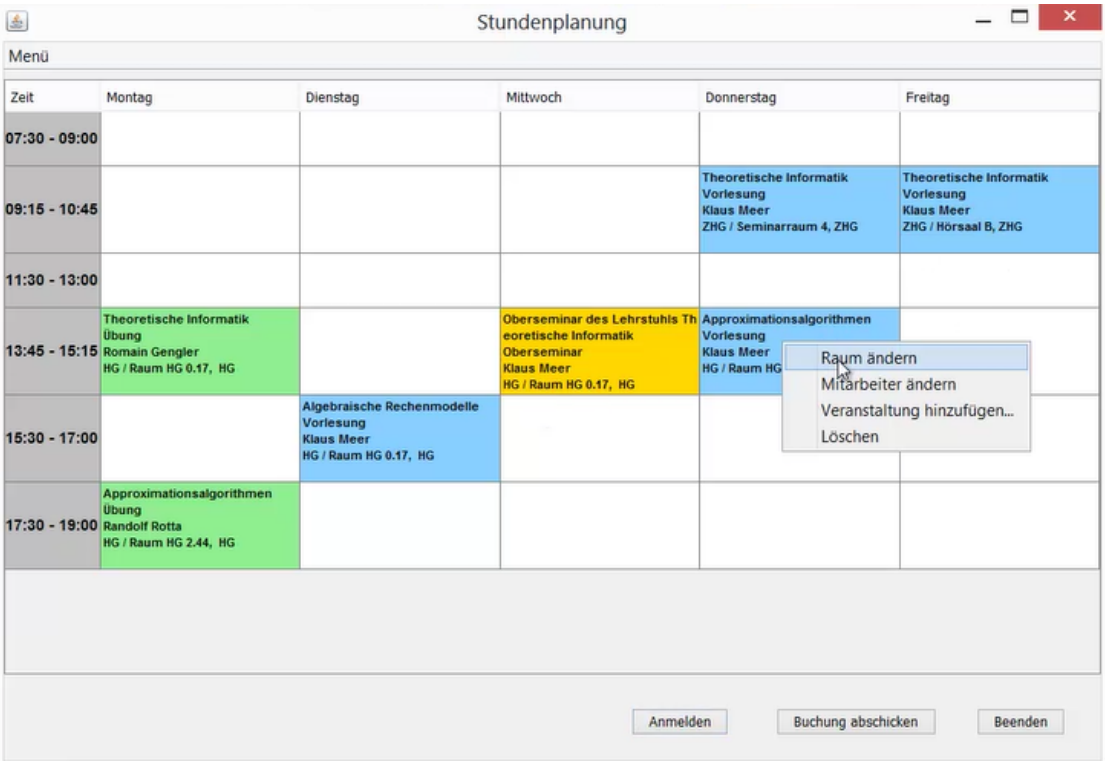


Abbildung 4.16: Buchungsansicht für einen Lehrstuhl

Manuelle Nachbearbeitung des Plans

Die Dozenten haben die Möglichkeit, individuelle Buchungswünsche für fakultative Lehrveranstaltungen im Planungssystem zu hinterlegen. Abbildung 4.16 zeigt die individuelle Buchungsansicht für einen Lehrstuhl. Der dargestellte Wochenplan enthält alle Lehrveranstaltungen, für die dieser Lehrstuhl in einem bestimmten Semester verantwortlich ist. Für jede Veranstaltung können die gewünschten Räume, Blöcke und Wochentage angegeben werden. Durch einen Klick auf den Button „Buchung abschicken“ werden die Buchungswünsche zentral abgespeichert und sind somit für den verantwortlichen Planer sichtbar.
Über den Menüeintrag „Server“ kann der Planer die Buchungswünsche der Dozenten aufrufen (Abbildung 4.17). Sie werden bei der automatischen Planerstellung nicht berücksichtigt, sondern dienen als Vorlage für eine manuelle Anpassung des Plans. Zur Umsetzung von Buchungswünschen und anderen Planänderungen kann eine zeitliche Verschiebung von Kursen mit Drag-and-Drop direkt im Stundenplan vorgenommen werden (Abbildung 4.14). Alle Eigenschaften einer Verstaltung können auch in einem Eigenschaftendialog bearbeitet werden, der über das Kontext-

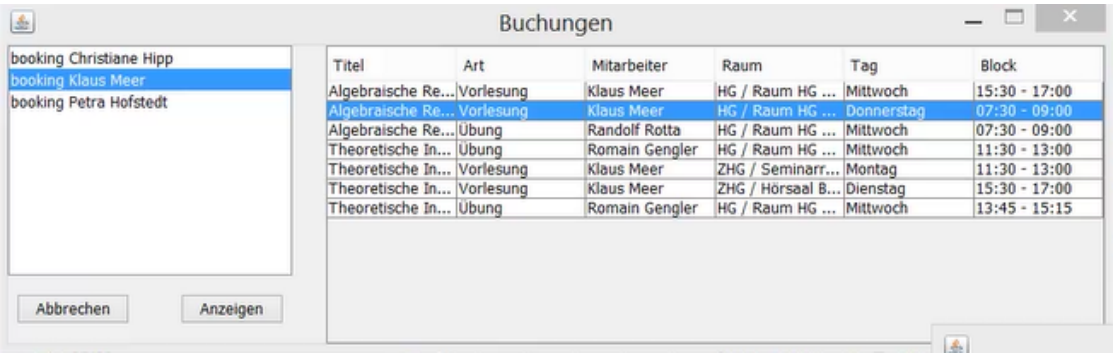


Abbildung 4.17: Ansicht der Buchungswünsche für den Planer

menü aufgerufen werden kann (Abbildung 4.18). Nach jeder manuellen Änderung über Drag-and-Drop oder über den Eigenschaftsdialog erfolgt eine automatische Anpassung des Plans, um alle Randbedingungen zu erfüllen. Eine Zuweisung von gesperrten Räumen und Dozenten oder von Räumen, die nicht die erforderliche Kapazität besitzen, wird automatisch rückgängig gemacht.

4.2.2 Entwicklung eines Interaktionskonzeptes unter Anwendung der Richtlinien

Das Stundenplanungssystem bietet eine automatische Planerstellung, die sich positiv auf die Benutzererfahrung auswirkt, da sie den Planer vom Aufwand der manuellen Planung befreit. Auch die Funktionen zur manuellen Planänderung können in Bezug auf die Benutzererfahrung positiv bewertet werden, da sie es dem Planer erlauben, sein Expertenwissen einzubringen. In den folgenden Abschnitten wird gezeigt, wie die Benutzererfahrung mit Hilfe der Interaktionsrichtlinien noch weiter verbessert werden kann. Zunächst werden Empfehlungen zur Verbesserung der regel- und wissensbasierten Planung gegeben. In den nachfolgenden Abschnitten wird ein vollständiges Interaktionskonzept für die Unterstützung der wissensbasierten Planung beschrieben.

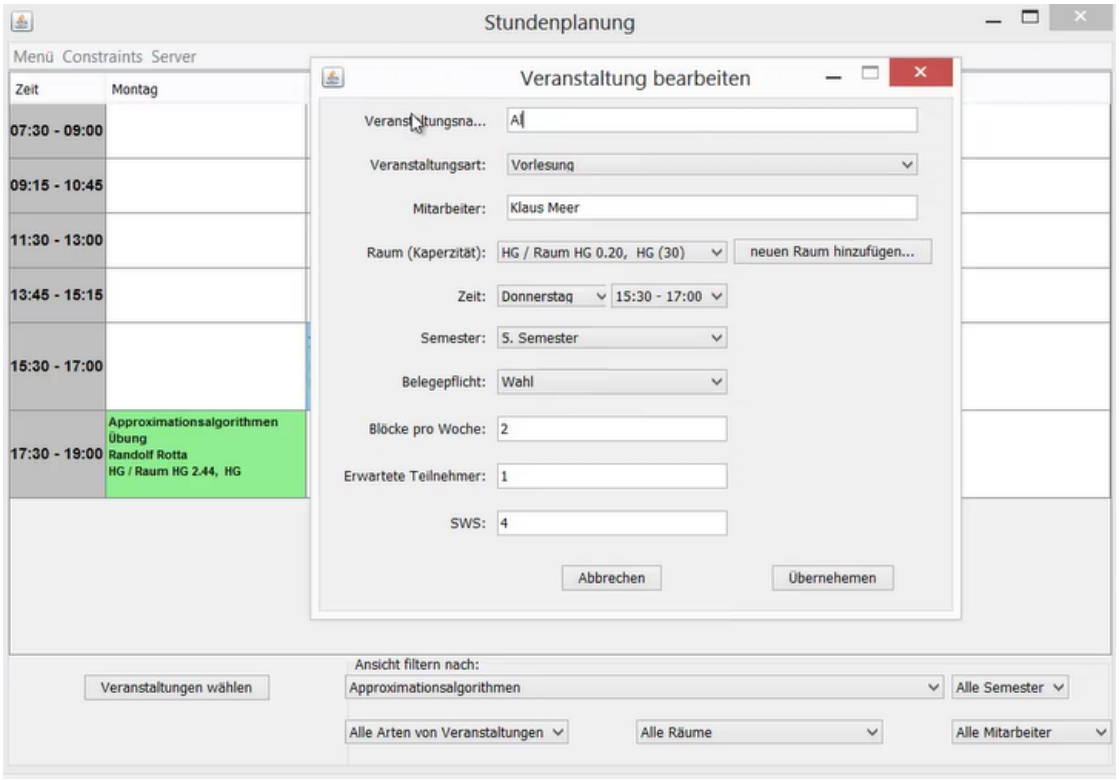


Abbildung 4.18: Filterung des Stundenplans nach der Lehrveranstaltung „Approximationsalgorithmen“. Änderung der Kurseigenschaften, um den Buchungswünschen zu entsprechen.

Unterstützung der regelbasierten und wissensbasierten Planung

Die Umsetzung von Buchungswünschen kann als Bestandteil der regelbasierten Planung aufgefasst werden. Diese Aufgabe sollte vollständig automatisiert werden, um den Planer von der aufwändigen manuellen Planungsarbeit zu entlasten. Zu diesem Zweck sollte das Planungssystem in der Lage sein, alle Buchungswünsche automatisch einzulesen und als empfohlene Randbedingungen zu berücksichtigen. Außerdem sollte die automatische Planung von einer Zielfunktion gesteuert werden, die nach einer möglichst geringen Anzahl von Abweichungen gegenüber den Buchungswünschen strebt.
Darüber hinaus sollte das Planungssystem die Einbringung von zusätzlichem Wissen im Rahmen einer wissensbasierten Planung unterstützen. Es bietet bereits die Möglichkeit, für einzelne Kurse manuelle Änderungen vorzunehmen und nimmt eine automatische Anpassung des restlichen Plans vor. Dabei wird nur der gerade betrachtete Kurs automatisch fixiert, sodass u.U. einige der in vorhergehenden

Schritten manuell vorgenommenen Modifikationen wieder aufgehoben werden. Dadurch werden u.U. bereits umgesetzte Buchungswünsche verletzt. Dieser Nachteil kann behoben werden, indem eine Möglichkeit zur Fixierung von Kurseigenschaften bereitgestellt wird.
Die Entscheidung, welche Eigenschaften einer Fixierung unterzogen werden sollen, ist für den Planer jedoch u.U. schwierig, wenn nicht bekannt ist, wie stark die Dozenten ihre jeweiligen Buchungen für Räume, Tage und Blöcke favorisieren. Als Orientierungshilfe für den Planer könnte daher für Dozenten eine Möglichkeit bereitgestellt werden, ihre Buchungswünsche zu priorisieren. Es ist zu erwarten, dass eine Verletzung eines Wunsches mit niedrigerer Priorität besser akzeptiert wird als die Verletzung eines Wunsches mit höherer Priorität.
Es wird empfohlen, die Festlegung von mindestens zwei Prioritätsstufen bei der Angabe der Buchungswünsche für Räume, Wochentage und Blöcke eines Kurses zu ermöglichen.

Beispiel 4.1: Ein Dozent möchte seine Lehrveranstaltungen gern stets im ersten oder zweiten Block abhalten, um den Nachmittag für seine Forschungstätigkeiten frei zu haben. Der konkrete Wochentag und der Raum sind dagegen für ihn zweitrangig. Daher weist er der Blockzuweisung die höchste Priorität zu.

Ermittlung der Zuweisungsregel nach Richtlinie R1

Für die Weiterentwicklung des bestehenden Systems ist es zunächst erforderlich, eine Zuweisungsregel festzulegen, die dem System zugrunde gelegt werden soll. Die in Abbildung 3.8 gezeigte Zuweisungsregel (2,3,4) wird als geeignet betrachtet, da sie die typischen Beziehungen zwischen den Attributen Wochentag, Block und Raum in einem Stundenplan widerspiegelt (Beispiele 3.6 und 3.9, Abschnitt 2.2.3). Aus der Zuweisungsregel ergeben sich 4 Abstraktionsebenen, die zur Konfiguration der Planungsfunktionen (Richtlinie R6) und der Konfliktauflösung (Richtlinie R8) benötigt werden. Abweichende Zuweisungsregeln für einzelne Kurse werden, wie in den folgenden Abschnitten gezeigt wird, durch die Bereitstellung entsprechender Fixierungsmöglichkeiten unterstützt.

Manuelle Anpassung von Plänen nach Richtlinie R2

Richtlinie R2 wird vom Planungssystem bereits im Wesentlichen erfüllt, da der Planer jede Eigenschaft eines Kurses manuell bearbeiten kann. Wenn die Änderung einen Konflikt verursacht, wird jedoch eine automatische Umplanung ausgelöst. Eine Beurteilung des Zustands des Plans, die zu einem Erkenntnisgewinn über das Planungsmodell und zu Schlussfolgerungen über weitere Planungsmaßnahmen führt, ist in diesem Fall nicht möglich. Es wird daher empfohlen, auftretende Konflikte zunächst nach R3 zu kennzeichnen und automatische Maßnahmen zur Konfliktauflösung erst nach Aufforderung des Planers vorzunehmen.

Überprüfung der Einhaltung von Randbedingungen nach Richtlinie R3

Bei der manuellen Planung kann eine Verletzung von Randbedingungen auftreten. Meist handelt es sich um zeitliche Überlappungen zwischen Kursen, die nicht im selben Raum oder aufgrund der Studienordnungen nicht zur selben Zeit stattfinden dürfen. In der Regel gibt es mehrere Möglichkeiten für eine Umplanung, um die Gültigkeit des Plans wiederherzustellen. Diese können mit den im nachfolgenden Abschnitt beschriebenen Fixierungen und Planungsfunktionen umgesetzt werden. Für welche Möglichkeit sich der Disponent entscheidet, hängt u.a. von der Art der Randbedingung, den betroffenen Kursen und deren Prioritätsvergabe ab. Der Planer sollte nach Richtlinie R3 über diese Fakten informiert werden. Eine automatisch angestoßene Umplanung nach einer vorgenommenen Modifikation ist daher nicht empfehlenswert.

Beispiel 4.2: Der Planer verschiebt einen Kurs in einen anderen Block. Dort ist jedoch bereits ein zweiter Kurs eingeplant, der im selben Raum stattfinden soll. Der Planer kann u.a. folgende Umplanungsmöglichkeiten in Betracht ziehen:

1. Der zweite Kurs wird in einen anderen Block am gleichen Tag verschoben.
2. Der zweite Kurs wird in einen Block an einem anderen Tag verschoben.
3. Beide Kurse finden zum gleichen Zeitpunkt statt, einer der beiden Kurse wird in einen anderen Raum verlegt.

Bei beiden Kursen besitzen die Block- und Tagzuweisungen die höchste Priorität. Daher entscheidet sich der Planer dafür, den Konflikt mit Variante 3 aufzulösen.

Überprüfung der Einhaltung strukturierter Anforderungen nach Richtlinie R4

Bei der Stundenplanung, die unter der Aufsicht des Planers auf der Basis des Vorjahresplans durchgeführt wird, besteht die strukturierte Anforderung darin, möglichst viele Buchungswünsche zu erfüllen. Nach Richtlinie R4 sollte die Einhaltung dieser Anforderung automatisch überprüft werden. Als Kennzahl kann dafür z.B. der Anteil der erfüllten Buchungswünsche an der Gesamtzahl der Buchungswünsche berechnet und dem Planer zur Orientierung präsentiert werden.
Auch die Prioritäten der Buchungswünsche können in die Berechnung von Kennzahlen einbezogen werden. Sie ermöglichen eine differenziertere Einschätzung der Planungsqualität. Es ist z.B. denkbar, den Anteil der erfüllten Buchungswünsche für jede Prioritätsstufe einzeln zu berechnen. Anhand dieser drei Kennzahlen kann der Planer einschätzen, ob die höher priorisierten Buchungswünsche ausreichend berücksichtigt werden. Ein geringer Anteil niedrig priorisierter Wünsche wird u.U. zugunsten der höher priorisierten Wünsche akzeptiert.

Explizite Fixierung nach Richtlinie R5 (Werkzeug I)

Um einen Plan zu erhalten, der alle Randbedingungen erfüllt, kann es erforderlich sein, von einigen Buchungswünschen abzuweichen. Der Disponent sollte nach Richtlinie R5 die Möglichkeit erhalten, durch Fixierung die Abweichung von einzelnen Buchungswünschen zu verhindern. Es wird empfohlen, für jeden Kurs eine getrennte Fixierung von Wochentag, Block und Raum zu ermöglichen.
Bei der Wahl der Fixierungen kann sich der Planer an den Prioritäten der Buchungswünsche orientieren. Gleichzeitig erhält er die Möglichkeit, durch die Wahl entsprechender Fixierungen die Zuweisungsregel eines Kurses zu beeinflussen oder die vergebenen Prioritäten vorübergehend außer Kraft zu setzen. Dies kann bei der Umplanung oder bei der Auflösung von Planungskonflikten nützlich sein.

Beispiel 4.3: Der Planer möchte einige zusätzliche Kurse, die verspätet angemeldet wurden, in den bestehenden Plan integrieren. Die bisherigen, bereits veröffentlichten Stundenpläne der Dozenten sollen dabei so wenig wie möglich verändert werden. Der Planer beschließt daher, die Wochentage der eingeplanten Kurse zu fixieren, auch wenn bei bestimmten Kursen der Wochentag die niedrigste Priorität besitzt.

Die Effizienz der Fixierung kann erhöht werden, indem die Möglichkeit bereitgestellt wird, ein bestimmtes Attribut bei allen oder bei einer Gruppe ausgewählter Kurse auf Knopfdruck zu fixieren. Es wird empfohlen, eine Gruppenfixierung von Wochentag, Block und Raum für jede Gruppe von Kursen, die einem gemeinsamen Wochentag zugeordnet sind, zu ermöglichen.
Eine weitere nützliche Funktion ist eine *bedingte Fixierung* einer Gruppe von Kursen, bei der ein bestimmtes Attribut nur dann fixiert wird, wenn es eine bestimmte Priorität besitzt. Auf diese Weise kann sichergestellt werden, dass die jeweiligen Prioritäten auch bei einer Gruppenfixierung nicht außer Kraft gesetzt werden. Es wird empfohlen, zu jeder Funktion der Gruppenfixierung zusätzlich eine Funktion der bedingten Fixierung bereitzustellen, bei der nur Attribute mit der höchsten Priorität fixiert werden.
Die im bestehenden System vorhandene Möglichkeit zum Filtern der Ansicht nach bestimmten Kurseigenschaften (vgl. 4.14) sollte beibehalten werden. Die Gruppenfixierung sollte nur auf die gerade angezeigten, nicht aber auf die ausgeblendeten Kurse angewendet werden. Auf diese Weise erhält der Planer eine zusätzliche Möglichkeit, eine Gruppenfixierung nach bestimmten Kriterien effizient umzusetzen.

Planungsfunktionen nach Richtlinie R7 (Werkzeug III)

Planungsfunktionen helfen dabei, Konflikte aufzulösen (vgl. Beispiel 4.4), die Umplanung eines bestehenden Plans zu steuern und Entwurfsentscheidungen für einen neuen Plan umzusetzen. Mögliche Planungsfunktionen für die Stundenplanung wurden bereits in Abschnitt 3.6 vorgestellt (Beispiel 3.9). Es wird empfohlen, die Planungsfunktionen „Full Scheduling“ (FS) und „Day Scheduling 2“ (DS2) und „Block Scheduling“ (BS) umzusetzen. Diese können mit Hilfe der expliziten Fixierung auch verwendet werden, um die restlichen Planungsfunktionen „Day

Scheduling“ (DS) und „Room Scheduling“ (RS) zu realisieren:

DS:
1. Fixierung der Räume bei allen Kursen, die dem ausgewählten Raum am ausgewählten Wochentag zugeordnet sind.
2. Fixierung aller Attribute bei den restlichen Kursen, die am ausgewählten Wochentag stattfinden.
3. Anwendung von DS2 für den ausgewählten Wochentag.

RS:
1. Fixierung des Raums bei allen Kursen, die dem ausgewählten Raum zugeordnet sind.
2. Fixierung aller Attribute bei den restlichen Kursen des Plans.
3. Anwendung von FS.

Um die Berücksichtigung der Prioritäten zu erleichtern, kann FS um zwei weitere Planungsfunktionen ergänzt werden:

FS(P1P2): FS mit automatischer Fixierung der Attribute mit der höchsten und zweit-höchsten Priorität.

FS(P1): FS mit automatischer Fixierung der Attribute mit der höchsten Priorität.

Wenn mit einer bestimmten Planungsfunktion keine Lösung gefunden werden kann, sollte der Planer durch eine Fehlermeldung darüber informiert werden. Anschließend sollte der ursprüngliche Zustand des Plans wiederhergestellt werden.
Die Planungsfunktionen sollten sich stets auf den vollständigen Plan beziehen. Auch die Kurse, die in der gefilterten Ansicht ggf. nicht sichtbar sind, sollten in die Planung einbezogen werden.

Konfliktauflösung nach Richtlinie R9 (Werkzeug V)

Richtlinie R9 lässt sich zur Auflösung von Planungskonflikten unter Berücksichtigung der Zuweisungsregel einsetzen. Sie bewirkt, dass der aktuelle Zustand des Plans so weit wie möglich beibehalten wird. Es wird empfohlen, für jede Zuweisung, die eine Randbedingung verletzt, eine automatische Funktion zur Konfliktauflösung bereitzustellen:

- Bei einer ungültigen Raumzuweisung sollte zunächst die Funktion „Block Scheduling“ eingesetzt werden, um den Konflikt auf Abstraktionsebene 1 aufzulösen (vgl. Abbildung 3.11 rechts). Die weitestmögliche Beibehaltung der Raumzuweisungen für die Kurse im Block sollte angestrebt werden. Wenn eine gültige Raumzuweisung nicht möglich ist, sollte nach Richtlinie R9 der Konflikt auf Abstraktionsebene 2 aufgelöst werden bzw. wenn nötig, auf höheren Ebenen.

- Bei einer ungültigen Blockzuweisung sollte der Konflikt nach Richtlinie R9 auf Abstraktionsebene 2 aufgelöst werden. Dabei wird die Beibehaltung der aktuellen Blockzuweisungen für alle Kurse angestrebt. Wenn nötig, muss der Konflikt auf höheren Ebenen aufgelöst werden.

- Bei einer ungültigen Tageszuweisung sollte der Konflikt nach Richtlinie R9 auf Ebene 3 aufgelöst werden.

In der Regel sind mindestens zwei Kurse, die am selben Wochentag stattfinden sollen, an der Verletzung einer Randbedingung beteiligt. Der Planer möchte ggf. mitbestimmen, auf welche Weise der Konflikt aufgelöst wird. Zu diesem Zweck sollte das Planungssystem berücksichtigen, für welchen Kurs die Konfliktauflösungsfunktion aufgerufen wird: Der aufgerufene Kurs sollte vorrangig verschoben werden, die restlichen Kurse sollten vorrangig ihre aktuelle Position beibehalten (vgl. Beispiel 4.4).

Beispiel 4.4: Zwei Kurse K_1 und K_2 finden zum gleichen Zeitpunkt (Montag, Block 3) statt, obwohl beide Pflichtveranstaltungen für die gleiche Seminargruppe sind. Der Planer ruft die Funktion zur Konfliktauflösung für den Block von K_1 auf, da er K_2 in Block 3 belassen möchte. Aufgrund der Zuweisungsregel versucht das Planungssystem zunächst, K_1 in einen anderen Block am selben Tag zu verschieben (Abstraktionsebene 3). Es gelingt, K_1 in Block 4 zu verschieben und einen passenden Raum zu finden.[12].

[12] Weitere, in anderen Wochentagen vorliegende Konflikte können zunächst erhalten bleiben. Wenn jedoch eine Auflösung des Konfliktes für K_1 nicht innerhalb des gleichen Wochentages möglich ist, erfolgt die Auflösung auf der nächst höheren Ebene, d.h. für die ganze Woche. Dabei müssen auch alle weiteren Konflikte mit aufgelöst werden.

Unterstützung bei der Wertauswahl nach den Richtlinien R6 und R8 (Werkzeug II und IV)

Das bestehende Planungssystem ermöglicht es einzelnen Kursen Räume, Blöcke und Wochentage zuzuweisen. Im Drop-Down-Menü eines Eigenschaftendialogs werden dabei zunächst alle verfügbaren Räume, Blöcke und Wochentage als Werte vorgeschlagen (z.B. in Abbildung 4.18). Die Gültigkeit der Wertzuweisungen wird jedoch erst nach der Zuweisung vom System überprüft. Ungültige Zuweisungen werden dabei rückgängig gemacht. Dies erhöht den manuellen Arbeitsaufwand und verschlechtert damit die Benutzererfahrung. Nach Richtlinie R6 sollten nur Werte aus dem Wertebereich einer Variablen für die Zuweisung vorgeschlagen werden. So sollten z.B. nach der Auswahl eines bestimmten Raums nur Blöcke vorgeschlagen werden, in denen dieser Raum nicht gesperrt ist. Außerdem sollte das Planungssystem diejenigen Wochentage, Blöcke und Räume hervorheben, deren Zuweisung keine nachträgliche Umplanung von anderen Kursen erfordert.
Beim Verschieben eines Kurses mit Drag-and-Drop in der Plantafel werden Blöcke rot markiert, die für den Kurs ungültig sind, da zu diesem Zeitpunkt bereits ein Kurs eingeplant ist, der sich mit dem gerade betrachteten Kurs nicht überlappen darf (dies entspricht Richtlinie R6). Eine Zuweisung zu einem rot markierten Block wird jedoch automatisch wieder rückgängig gemacht. Dies verschlechtert die Benutzererfahrung, da der Planer nicht selbst entscheiden kann, auf welche Weise der Konflikt aufgelöst wird. Es wird empfohlen, die fehlerhafte Zuweisung zu ermöglichen und anschließend alle am Konflikt beteiligten Kurse hervorzuheben, sodass der Planer entscheiden kann, welcher Kurs verschoben wird (vgl. Beispiel 4.4).
Auch das Ausmaß der erforderlichen Umplanung kann die Entscheidung des Planers für einen bestimmten Wert beeinflussen. Nach R8 sollte das Planungssystem für jeden Wert der ausgewählten Variablen ermitteln, auf welcher Abstraktionsebene die Umplanung stattfinden muss. Der Planer erfährt dadurch,

- ob eine Raumänderung ausreicht (AE 0)
- ob eine Umplanung innerhalb desselben Wochentags ausreicht (AE 1) oder
- ob die Wochentage von ein oder mehreren Kursen geändert werden müssen (AE 2)

um einen bestimmten Wert umzusetzen (vgl. Beispiel 3.9). Zuweisungen, die sich

mit den gegebenen Kursen und Randbedingungen nicht in einem gültigen Plan umsetzen lassen, sollten ebenfalls entsprechend gekennzeichnet werden.
Auch Kurse, die in der Plantafel durch eine Filterung ausgeblendet wurden, sollten bei der automatischen Ermittlung und Kennzeichnung von Zuweisungsvorschlägen berücksichtigt werden.

4.2.3 Entwurf einer Benutzeroberfläche zur Umsetzung des Interaktionskonzeptes

Im Folgenden wird anhand des in Abbildung 4.19 abgebildeten Prototyps demonstriert, wie das im vorigen Abschnitt beschriebene Interaktionskonzept in einer grafischen Benutzeroberfläche umgesetzt werden kann.

Visualisierung von Buchungswünschen, Planungsfehlern und Kennzahlen

Eine Verletzung von Buchungswünschen wird durch eine entsprechende Farbgebung der Buttons in der Buttonleiste eines Kurses erkenntlich gemacht. Die rote Einfärbung eines Attributes zeigt an, dass der ihm zugeordnete Buchungswunsch für diesen Kurs die höchste Priorität besitzt und von der aktuellen Zuweisung verletzt wird (z.B. Dienstag, 2. Block, HG 0.17). Eine gelbe Einfärbung zeigt an, dass ein Buchungswunsch mit einer niedrigeren Priorität verletzt wird (z.B. Donnerstag, 1. Block, 7:30-9:00). Wenn ein Button mit der Maus überfahren wird, erscheint rechts außen ein kleiner Pfeil (z.B. Mittwoch, 1. Block, LG 1A/121). Wird dieser mit der Maus überfahren, öffnet sich ein Drop-Down-Menü, in dem Werte aufgelistet werden, die für das Attribut ausgewählt werden können. An der Spitze dieser Auflistung erscheinen die nach Prioritäten gekennzeichneten Buchungswünsche. „P1“ steht dabei für die höchste Priorität, „P2“ und „P3“ für die niedrigeren Prioritäten. Für die restlichen Einträge der Auflistungen existieren keine Buchungswünsche.
Die Attribute, deren Werte die Ursache für die Verletzung einer Randbedingung sind, werden mit einem roten Rahmen gekennzeichnet (z.B. Dienstag, 2. Block, ZHG Audimax 1). Wenn z.B. zwei Kurse zur gleichen Zeit stattfinden, obwohl eine Überlappung aufgrund der Studienordnung nicht zulässig ist, werden die Blöcke beider Kurse rot gekennzeichnet. Wenn eine zeitliche Überlappung zulässig

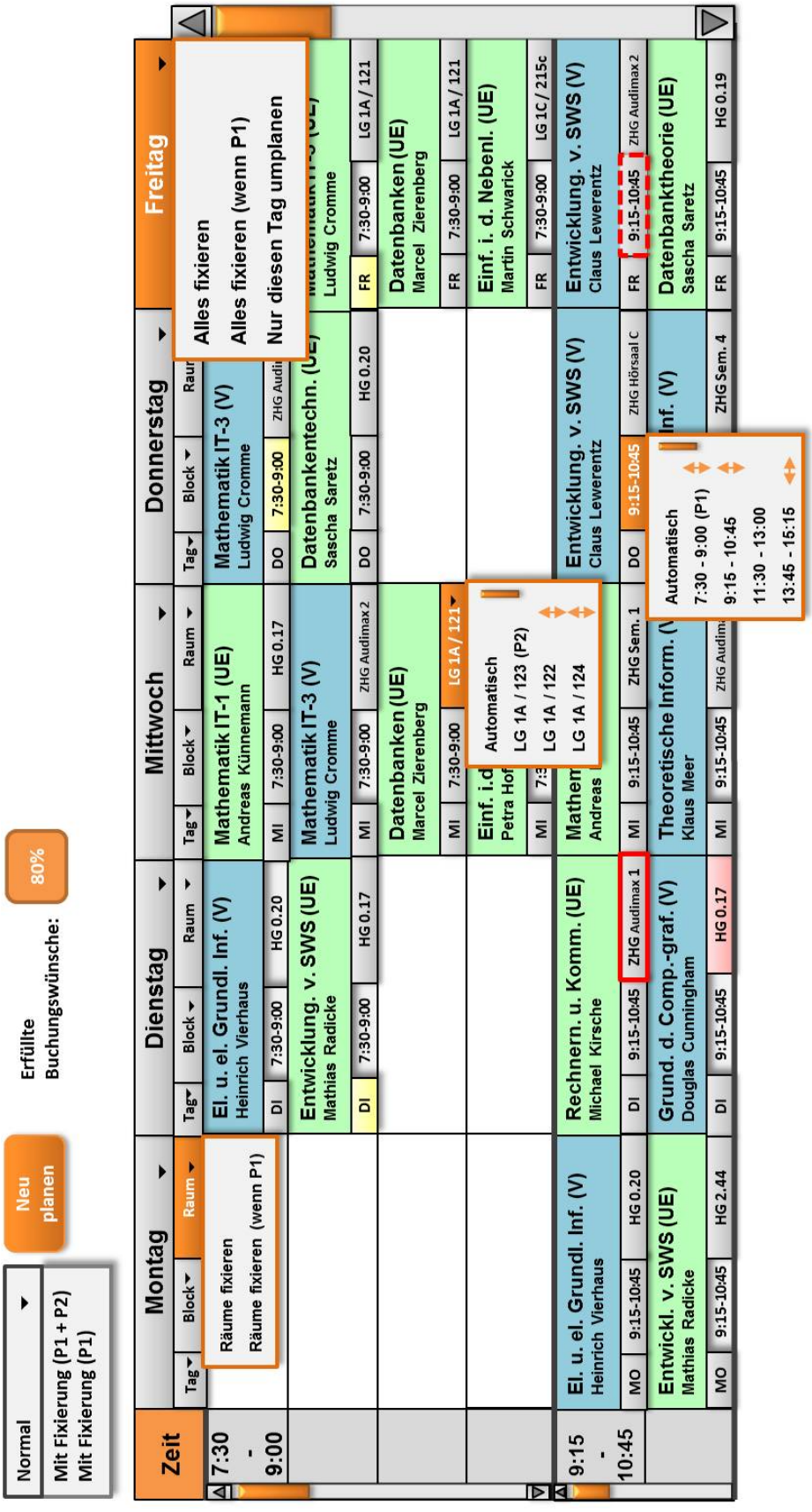


Abbildung 4.19: Entwurfsskizze einer Benutzeroberfläche, in der das Interaktionskonzept für die Stundenplanung umgesetzt wird.

ist, aber derselbe Raum gewählt wurde, werden die Räume beider Kurse rot gekennzeichnet (in Abbildung 4.19 ist der zweite am Konflikt beteiligte Kurs aus Platzgründen nicht sichtbar).
Bei der Verletzung einer empfohlenen Randbedingung werden die entsprechenden Attribute der betroffenen Kurse mit einer rot gestrichelten Linie umrandet (z.B. Freitag, 2. Block, 9:15-10:45).
Der Anteil der erfüllten Buchungswünsche an der Gesamtzahl der Buchungswünsche wird als Kennzahl oberhalb der Plantafel dargestellt.

Interaktionselemente zur Durchführung von Fixierungen

Der Zustand der Fixierung wird für Tag, Block und Raum eines Kurses direkt in der Plantafel erkenntlich gemacht werden. Um die Fixierungsart darzustellen, wurde jedem Feld in der Plantafel, das einen Kurs repräsentiert, eine graue Button-Leiste mit drei Buttons hinzugefügt, die jeweils mit dem Wochentag, dem Block und dem Raum des Kurses beschriftet sind. Ein Button wird optisch als „eingedrückt“ visualisiert, wenn das zugehörige Attribut fixiert ist. Die Anzeigeform als Buttons ist eine Affordanz, die den Planer (u.U. sogar ohne vorherige Einweisung) auf die erforderliche Aktion zur Fixierung (Eindrücken des Buttons) und zur Aufhebung der Fixierung (erneutes Drücken des Buttons) schließen lässt.
Um den Fixierungszustand eines Attributes zu wechseln, genügt es, den entsprechenden Button in der Buttonleiste des Kurses anzuklicken.
Der Kopf jeder Spalte eines Wochentages enthält ein dreiteiliges Menü, über welches sich Fixierungsoptionen für Tage, Blöcke und Räume aufrufen lassen. Für jedes Attribut wird eine normale und eine bedingte Gruppenfixierung angeboten, die auf die Kurse angewendet wird, die dem jeweiligen Tag zugeordnet sind. Auch eine gleichzeitige Fixierung von Tag, Block und Raum ist für alle Kurse eines bestimmten Tages möglich. Um diese Funktion aufzurufen, muss das Drop-Down-Menü neben der Beschriftung des Wochentages geöffnet werden.

Interaktionselemente zum Aufruf von Planungsfunktionen und zur Konfliktbeseitigung

Im Drop-Down-Menü eines Wochentages kann die Funktion „Nur diesen Tag umplanen“ aufgerufen werden. Sie entspricht der Planungsfunktion „Day Scheduling“

(DS2). Oberhalb der Plantafel kann die Funktion „Full Scheduling“ (FS) über den Button „Neu planen“ aufgerufen werden. Links von diesem Button befindet sich ein Drop-Down-Menü mit Fixierungsoptionen. Sie können ausgewählt werden, um statt FS die Funktionen FS(P1P2) oder FS(P1) auszuführen.
Im Drop-Down-Menü eines Attributes wird den vorgeschlagenen Werten die Option „Automatisch“ vorangestellt. Sie ist ausgegraut, d.h. deaktiviert, wenn das Attribut fixiert ist. Diese Option dient der automatischen Auflösung von Konflikten nach Richtlinie R9. Wenn keine Konflikte vorliegen, bleiben die aktuellen Eigenschaften der Kurse erhalten.

Visualisierung der Entscheidungsunterstützung bei der Wertauswahl

Die Änderung eines Wertes für ein bestimmtes Attribut kann auf zwei Wegen erfolgen:

- Verschieben des Kurses per Drag-and-Drop (anwendbar für Wochentag und Block). Die Kennzeichnung der Werte erfolgt direkt in der Plantafel.
- Auswahl eines Wertes aus dem Drop-Down-Menü (anwendbar für alle Attribute). Die Kennzeichnung der Werte erfolgt im Menü.

Die Kennzeichnung der zur Auswahl stehenden Werte erfolgt nach den Richtlinien R6 und R8:

- Werte, die ohne Umplanung anwendbar sind, erhalten keine Kennzeichnung.
- Werte, die nicht anwendbar sind, da sie nicht im Wertebereich der Variable enthalten sind, werden ausgegraut dargestellt und können nicht ausgewählt werden.
- Werte, für die eine Umplanung innerhalb des gleichen Wochentages möglich ist, werden mit einem vertikalen Pfeil gekennzeichnet.
- Werte, für die eine Umplanung notwendig ist, bei sich die Wochentage einiger Kurse ändern, werden mit einem horizontalen Pfeil gekennzeichnet.

4.2.4 Sonstige Empfehlungen

Im Planungssystem sollte eine Möglichkeit bereitgestellt werden, jede Aktion (Fixierung, Aufruf von Planungsfunktionen, manuelle Verschiebung von Kursen etc.) rückgängig zu machen bzw. wiederherzustellen. Auf diese Weise kann die Entwurfsphase, in der ggf. mehrere Entscheidungsalternativen miteinander verglichen werden, zusätzlich unterstützt werden.
Die Anwendung der Interaktionsrichtlinien sollte mit Blick auf die benötigte Rechenzeit für die automatische Ermittlung von Planungsvorschlägen abgewogen werden. Auf eine Kennzeichnung von Werten nach Richtlinie R8 sollte verzichtet werden, wenn die dafür erforderlichen Berechnungen mehrere Minuten in Anspruch nehmen.

5 Zusammenfassung

Es war das Ziel dieser Arbeit, ein allgemeingültiges Konzept der Mensch-Computer-Interaktion zu definieren, mit dem eine ausreichende Einbeziehung des menschlichen Expertenwissens in die Planerstellung sichergestellt werden kann. Zunächst wurden die Merkmale praktischer Planungsprobleme untersucht (Kapitel 2). Es wurden Gemeinsamkeiten in der Struktur von Produktions-, Flotten-, und Stundenplanungsproblemen sowie weiteren Planungsproblemen festgestellt (Abschnitt 2.2). Die Herausforderungen bei der automatischen Lösung und Optimierung von Planungsproblemen, die aus der kombinatorischen Vielfalt an Lösungsmöglichkeiten resultieren, wurden diskutiert (Abschnitt 2.3).
In der Praxis existieren häufig zusätzliche Anforderungen, die nicht im Planungsmodell abgebildet sind. Sie müssen vom Disponenten bei der Planung manuell umgesetzt werden. Es wurde untersucht, welche Entscheidungsprozesse dabei typischerweise ablaufen (Abschnitt 2.4). Auf dieser Grundlage konnte der interaktive Lösungsprozess für Planungsprobleme formal beschrieben werden, bei dem sich der Mensch mit herkömmlichen Interaktionswerkzeugen in den automatischen Lösungsprozess einschaltet (Abschnitt 3.1). Bei seiner Bewertung wurden Faktoren identifiziert, die den Aufwand der Planung für den Disponenten erhöhen: Aufgrund der Komplexität und Größe praktischer Probleme ist der Disponent in der Regel nicht in der Lage, die Zusammenhänge zwischen den Randbedingungen zu überblicken und die Optimierung der Zielfunktionen sicherzustellen. Daraus resultieren Planungsfehler, die aufwändig zu beseitigen sind oder suboptimale Pläne (Abschnitt 3.2).
Auf der Basis kooperativer Modelle für die Funktionsaufteilung (Abschnitt 3.3) wurde ein neues Interaktionskonzept entwickelt (Abschnitt 3.4), welches die Entscheidungsunterstützung für den Disponenten während der interaktiven Planung verbessern soll (Abschnitt 3.7). Das Konzept setzt sich aus 5 Interaktionswerkzeugen zusammen, die vom Disponenten dazu eingesetzt werden können, Entscheidungen zu treffen, Handlungsspielräume zu visualisieren und Konflikte aufzulösen. Die

Werkzeuge können für jedes Planungsproblem konfiguriert werden und sind unabhängig von der grafischen Gestaltung der Benutzeroberfläche (Abschnitt 3.6).
Die praktische Anwendung des Interaktionskonzepts wurde in 2 Fallstudien für Probleme der Stundenplanung und der Flottenplanung demonstriert (Kapitel 4). Anhand eines Anwendertests konnte nachgewiesen werden, dass mit dem Einsatz der Werkzeuge eine schnellere und erfolgreichere Bearbeitung praktischer Planungsprobleme moglich ist (Kapitel 4.1).
Die Ergebnisse dieser Arbeit resultieren aus einer interdisziplinären Betrachtungsweise, welche die theoretischen Grundlagen zur automatischen Lösung von Planungsproblemen mit der arbeitswissenschaftlichen Sichtweise auf die Aufgaben des Disponenten kombiniert. Sowohl Mensch als auch Computer sind Parteien, die mit unterschiedlichen Voraussetzungen und Zielstellungen am Lösungsprozess beteiligt sind. Nur eine ganzheitliche Betrachtung ihres Zusammenspiels führt zu Erkenntnissen über Defizite und Optimierungspotenzial bei der Computerunterstützung und letztlich zu geeigneten Maßnahmen zur Behebung dieser Defizite.

6 Ausblick

Die Interaktionsrichtlinien wurden entwickelt, um die Verfeinerung des Planungsmodells durch zusätzliche temporäre Randbedingungen zu erleichtern. Mit ihnen kann ein Unternehmen seine spezifischen Planungsbedingungen in automatisch generierten Plänen umsetzen. Eine Dispositionssoftware, die diese Richtlinien anwendet, kann daher auch von Unternehmen eingesetzt werden, deren Planungsproblem nicht exakt mit dem in der Software enthaltenen Planungsmodell übereinstimmt. Der Anwenderkreis der Software verbreitert sich somit. Die Voraussetzung dafür ist jedoch eine gute Gebrauchstauglichkeit der Benutzeroberfläche. Sie sollte im Idealfall selbstbeschreibend sein (Abschnitt 1.2.2). Das in dieser Arbeit entwickelte Interaktionskonzept eröffnet ein Forschungsgebiet, welches sich mit der gebrauchstauglichen Umsetzung der Interaktionsrichtlinien in der Benutzeroberfläche von Planungssystemen auseinandersetzt. Es ergeben sich u.a. folgende Fragestellungen:

- Wie können Planungsfunktionen und Fixierungsmöglichkeiten selbsterklärend in die Benutzeroberfläche integriert werden?
- Wie können die nach den Richtlinien R7 und R9 vom Computer generierten Planungsvorschläge so präsentiert werden, dass ihre korrekte Interpretation durch den Disponenten erleichtert wird?
- Welche Visualisierungsformen für Pläne sind geeignet, um das Verständnis der zugrunde liegenden Abstraktionsebenen zu fördern?
- Welche Visualisierungsformen für Pläne sind geeignet, um die Ableitung von unstrukturierten Bedingungen und Anforderungen aus dem aktuellen Zustand des Plans zu erleichtern?

Ein weiterer Forschungsgegenstand ist die Weiterentwicklung der bestehenden Computerarchitektur und der bestehenden Algorithmen, um die Performanz der auto-

matischen Planung zu steigern.

Literatur

Amilhastre, Jérôme, Fargier, Hélène und Marquis, Pierre (2002). „Consistency restoration and explanations in dynamic CSPs—Application to configuration“. In: *Artificial Intelligence* 135.1–2, S. 199–234.

Amoako-Gyampah, Kwasi (2007). „Perceived usefulness, user involvement and behavioral intention: an empirical study of ERP implementation“. In: *Computers in Human Behavior* 23.3, S. 1232–1248.

Anh, Duong Tuan, Tam, Vo Hoang und Hung, Nguyen Quoc Viet (2006). „Generating complete university course timetables by using local search methods“. In: *Research, Innovation and Vision for the Future, 2006 International Conference on*, S. 67–74.

– (2007). „Combined Interactive and Automatic Course Timetabling“. In: *Proceedings of International Workshop on Advanced Computing and Applications, ACOMP*, S. 14–16.

Artigues, C., Demassey, S. und Néron, E. (2013). *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. ISTE. Wiley.

Asprova (2013). „Asprova Online Help“. In: *Asprova Corporation [http://lib.asprova.com/onlinehelp/en/AS2003HELP00001000.html, abgerufen am 9.7.2017]*.

Ausiello, G. (1999). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-electronic-Media. U.S. Government Printing Office.

Bainbridge, L. (1997). „The change in concepts needed to account for human behavior in complex dynamic tasks“. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 27.3.

Baptiste, P., Pape, C.L. und Nuijten, W. (2001). *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. International Series in Operations Research & Management Science. Springer US.

Barták, Roman, Salido, Miguel A und Rossi, Francesca (2010). „New trends in constraint satisfaction, planning, and scheduling: a survey“. In: *The Knowledge Engineering Review* 25.03, S. 249–279.

Bayer, Martin (2011). „ERP-Trends“. In: *Computerwoche [http://www.computerwoche.de/a/erp-systeme-zu-langsam-fuer-das-business,2350345, abgerufen am 9.7.2017]*.

– (2012). „Modernes ERP - Belastungsprobe für die ERP-Architekturen“. In: *Computerwoche [http://www.computerwoche.de/a/belastungsprobe-fuer-die-erp-architekturen,2504509, abgerufen am 9.7.2017]*.

Bednárek, David u. a. (2010). „MetroNG: multimodal interactive scheduling interface“. In: *Proceedings of the International Conference on Advanced Visual Interfaces*. AVI '10. ACM, S. 317–320.

Begnaud, Joseph, Benjaafar, Saif und Miller, Lisa A. (2009). „The multi-level lot sizing problem with flexible production sequences“. In: *IIE Transactions* 41.8, S. 702–715.

Beldiceanu, Nicolas, Carlsson, Mats und Rampon, Jean-Xavier (2005). „Global constraint catalog“. In: *SICS Research Report, [http://sofdem.github.io/gccat/, abgerufen am 9.7.2017]*.

Berglund, Martina und Karltun, Johan (2007). „Human, technological and organizational aspects influencing the production scheduling process“. In: *International Journal of Production Economics* 110.1-2, S. 160–174.

Berrada, Ilham, Ferland, Jacques A und Michelon, Philippe (1996). „A multi-objective approach to nurse scheduling with both hard and soft constraints“. In: *Socio-Economic Planning Sciences* 30.3, S. 183–193.

Berry, Pauline, Peintner, Bart und Yorke-Smith, Neil (2007). „Bringing the User Back into Scheduling: Two Case Studies of Interaction with Intelligent Scheduling Assistants“. In: *Interaction Challenges for Intelligent Assistants*, S. 10–11.

Berry, Pauline M. u. a. (2005). „Mixed-initiative issues for a personalized time management assistant“. In: *Proceedings of ICAPS'05 Workshop on Mixed-Initiative Planning and Scheduling*, S. 12–17.

Bessiere, Christian (2006). „Constraint propagation“. In: *Foundations of Artificial Intelligence* 2, S. 29–83.

Bjorlin, Courtney (2010). „ERP usability and functionality: Must they be mutually exclusive?“ In: *SearchSAP [http://searchsap.techtarget.com/news/2240023066/ERP-usability-and-functionality-Must-they-be-mutually-exclusive, abgerufen am 9.7.2017]*.

– (2012). „SAP Restarts Its Usability Efforts With Start-Up Spirit“. In: *ASUG News [https://www.asug.com/news/sap-restarts-its-usability-efforts-with-start-up-spirit, abgerufen am 9.7.2017]*.

Brandenburg, U. u. a. (2013). *Marktspiegel Business Software ERP/PPS 2013/2014*. Forschungsinstitut für Rationalisierung (FIR) an der RWTH Aachen: TROVARIT AG.

Braun, Heinrich und Gruenwald, Claus (2000). „New Advances in SAP APO to Optimize the Supply Chain“. In: *SAP Insider [http://sapinsider.wispubs.com/Assets/Articles/2000/October/New-Advances-In-SAP-APO-To-Optimize-The-Supply-Chain, abgerufen am 9.7.2017]* 1.2.

Brucker, Peter und Knust, Sigrid (2001). „Resource-Constrained Project Scheduling and Timetabling“. In: *Practice and Theory of Automated Timetabling III*.

Hrsg. von Edmund Burke und Wilhelm Erben. Bd. 2079. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 277–293.

Brucker, Peter und Knust, Sigrid (2012a). „Algorithms and Complexity“. In: *Complex Scheduling*. GOR-Publications. Springer Berlin Heidelberg, S. 29–115.

– (2012b). „Scheduling Models“. In: *Complex Scheduling*. GOR-Publications. Springer Berlin Heidelberg, S. 1–28.

Bui, Lam Thu u. a. (2012). „Adaptation in Dynamic Environments: A Case Study in Mission Planning“. In: *Evolutionary Computation, IEEE Transactions on* 16.2, S. 190–209.

Burke, E. und Kendall, G. (2005). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. SpringerLink: Springer e-Books. Springer Science+Business Media, LLC.

Burke, Edmund K., Li, Jingpeng und Qu, Rong (2010). „A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems“. In: *European Journal of Operational Research* 203.2, S. 484–493.

Burke, Edmund K und Newall, James P. (1999). „A multistage evolutionary algorithm for the timetable problem“. In: *Evolutionary Computation, IEEE Transactions on* 3.1, S. 63–74.

Burke, Edmund K. u. a. (2004). „The State of the Art of Nurse Rostering“. In: *Journal of Scheduling* 7.6, S. 441–499. ISSN: 1094-6136.

Burstein, Frada und Holsapple, Clyde W. (2008). *Handbook on Decision Support Systems 1*. Springer-Verlag, Berlin Heidelberg.

Burstein, Mark H. u. a. (1996). „Issues in the development of human-computer mixed-initiative planning“. In: *Cognitive Technology*. Elsevier, S. 285–303.

Calisir, Fethi und Calisir, Ferah (2004). „The relation of interface usability characteristics, perceived usefulness, and perceived ease of use to end-user satisfaction with enterprise resource planning (ERP) systems“. In: *Computers in Human Behavior* 20.4, S. 505–515.

Campbell, Ann Melissa und Savelsbergh, Martin (2004). „Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems“. In: *Transportation Science* 38.3, S. 369–378.

Cegarra, Julien (2008). „A cognitive typology of scheduling situations: A contribution to laboratory and field studies“. In: *Theoretical Issues in Ergonomics Science* 9.3, S. 201–222.

Cegarra, Julien und Hoc, Jean-Michel (2008). „The role of algorithm and result comprehensibility of automated scheduling on complacency“. In: *Human Factors and Ergonomics in Manufacturing & Service Industries* 18.6, S. 603–620.

Cegarra, Julien und Wezel, Wout van (2011a). „A Comparison of Task Analysis Methods for Planning and Scheduling“. In: *Behavioral Operations in Planning and Scheduling*. Hrsg. von Jan C. Fransoo, Toni Waefler und John R. Wilson. Springer Berlin Heidelberg, S. 323–338.

– (2011b). „Allocating Functions to Human and Algorithm in Scheduling“. In: *Behavioral Operations in Planning and Scheduling*. Hrsg. von Jan C. Fransoo, Toni Waefler und John R. Wilson. Springer Berlin Heidelberg, S. 339–370.

– (2011c). „The Unsung Contribution of Production Planners and Schedulers at Production and Sales Interfaces“. In: *Behavioral Operations in Planning and Scheduling*. Hrsg. von Jan C. Fransoo, Toni Waefler und John R. Wilson. Springer Berlin Heidelberg, S. 47–81.

– (2012). „Revisiting Decision Support Systems for Cognitive Readiness: A Contribution to Unstructured and Complex Scheduling Situations“. In: *Journal of Cognitive Engineering and Decision Making* 6.3, S. 299–324.

Cheng, Peter C.-H. u. a. (2002). „Opening the Information Bottleneck in Complex Scheduling Problems with a Novel Representation: STARK Diagrams“. In: *Proceedings of the Second International Conference on Diagrammatic Representation and Inference*. DIAGRAMS '02. London, UK: Springer-Verlag, S. 264–278.

Cordeau, Jean-François u. a. (2007). „Vehicle Routing“. In: *Handbooks in Operations Research and Management Science* 14, S. 367–428.

Cormen, Thomas H (2010). *Algorithmen - Eine Einführung*. Oldenbourg Verlag.

Crawford, S. u. a. (1999). „Investigating the Work of Industrial Schedulers through Field Study“. In: *Cognition, Technology and Work* 1 (2), S. 63–77.

Cummings, M. L. (2004). „Automation Bias in Intelligent Time Critical Decision Support Systems“. In: *AIAA 3rd Intelligent Systems Conference*. AIAA, S. 2004–6313.

Dadashi, Nastaran u. a. (2013). „Practical use of work analysis to support rail electrical control rooms: A case of alarm handling“. In: *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit* 227.2, S. 148–160.

Davis, Fred D. (1989). „Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology“. In: *MIS Q.* 13.3, S. 319–340.

– (1993). „User acceptance of information technology: system characteristics, user perceptions and behavioral impacts“. In: *International Journal of Man-Machine Studies* 38.3, S. 475–487.

Davis, Matthew C. u. a. (2014). „Advancing socio-technical systems thinking: A call for bravery“. In: *Applied Ergonomics* 45.2, Part A, S. 171–180.

De Backer, Bruno u. a. (2000). „Solving vehicle routing problems using constraint programming and metaheuristics“. In: *Journal of Heuristics* 6.4, S. 501–523.

Dechter, R. (2003). *Constraint Processing*. The Morgan Kaufmann Series in Artificial Intelligence. Elsevier Science.

Dechter, Rina und Pearl, Judea (1989). „Tree clustering for constraint networks“. In: *Artificial Intelligence* 38.3, S. 353–366.

Dekker, S. W. A. und Woods, D. D. (2002). „MABA-MABA or Abracadabra? Progress on Human–Automation Co-ordination“. In: *Cognition, Technology & Work* 4 (4), S. 240–244.

Dillan, Steve (2011). „Evaluierung von Constraint-basierten Modellen zum Einsatz in Entscheidungsunterstützungssystemen am Beispiel von VRP“. Masterarbeit. Hochschule Zittau/Görlitz.

Dominic, P. D. D., Kaliyamoorthy, S. und Kumar, M. Saravana (2004). „Efficient dispatching rules for dynamic job shop scheduling“. In: *The International Journal of Advanced Manufacturing Technology* 24 (1), S. 70–75.

Durso, F.T. u. a. (2007). „Industrial Systems“. In: *Handbook of Applied Cognition*. John Wiley & Sons.

Ehrgott, M. (2005). *Multicriteria Optimization*. Lecture notes in economics and mathematical systems. Springer.

Endsley, Mica R. (1999). „Level of automation effects on performance, situation awareness and workload in a dynamic control task“. In: *Ergonomics* 42.3, S. 462–492.

Ernst, A.T u. a. (2004). „Staff scheduling and rostering: A review of applications, methods and models“. In: *European Journal of Operational Research* 153.1, S. 3–27.

Fagerholt, Kjetil (2004). „A computer-based decision support system for vessel fleet scheduling—experience and future research“. In: *Decision Support Systems* 37.1, S. 35–47.

Fargier, Hélène und Vilarem, Marie-Catherine (2004). „Compiling CSPs into tree-driven automata for interactive solving“. In: *Constraints* 9.4, S. 263–287.

Fehn, Andreas (2014). „Entwicklung einer interaktiven, graphischen Benutzeroberfläche für den constraint-basierten Konfigurator RangeConfig“. Bachelorarbeit. Brandenburgische Technische Universität Cottbus.

Fischer, Andreas, Greiff, Samuel und Funke, Joachim (2012). „The Process of Solving Complex Problems“. In: *The Journal of Problem Solving* 4.1, S. 19–24.

Framinan, Jose M. und Ruiz, Rubén (2010). „Architecture of manufacturing scheduling systems: Literature review and an integrated proposal“. In: *European Journal of Operational Research* 205.2, S. 237–246.

– (2012). „Guidelines for the deployment and implementation of manufacturing scheduling systems“. In: *International Journal of Production Research* 50.7, S. 1799–1812.

Frayman, Felix (2001). „User-interaction requirements and its implications for efficient implementations of interactive constraint satisfaction systems“. In: *Proc. CP 2001 workshop on user-interaction in constraint satisfaction, Paphos, Cyprus.*

Freuder, Eugene C. (1982). „A Sufficient Condition for Backtrack-Free Search“. In: *Journal of the ACM* 1, S. 24–32.

Freuder, EugeneC. und O'Sullivan, Barry (2001). „Generating Tradeoffs for Interactive Constraint-Based Configuration“. In: *Principles and Practice of Constraint Programming — CP 2001.* Hrsg. von Toby Walsh. Bd. 2239. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 590–594.

Funke, Birger, Grünert, Tore und Irnich, Stefan (2005). „Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration“. In: *Journal of Heuristics* 11.4, S. 267–306.

Gacias, Bernat, Cegarra, Julien und Lopez, Pierre (2012). „Scheduler-oriented algorithms to improve human-machine cooperation in transportation scheduling support systems“. In: *Engineering Applications of Artificial Intelligence* 25.4, S. 801–813.

Galer, Susan (2012). „Get Ready for a Design Revolution“. In: *SAP.info [http://en.sap.info/tired-of-sap-gui-talk-to-sam/82864, abgerufen am 9.7.2017].*

Garača, Željko (2011). „Factors related to the intended use of ERP systems“. In: *Management-Journal of Contemporary Management Issues* 2, S. 23–42.

Gasser, Roland, Fischer, Katrin und Wäfler, Toni (2011). „Decision Making in Planning and Scheduling: A Field Study of Planning Behaviour in Manufacturing“. In: *Behavioral Operations in Planning and Scheduling*. Hrsg. von Jan C. Fransoo, Toni Wäfler und John R. Wilson. Springer Berlin Heidelberg, S. 11–30.

Gayialis, Sotiris P. und Tatsiopoulos, Ilias P. (2004). „Design of an IT-driven decision support system for vehicle routing and scheduling“. In: *European Journal of Operational Research* 152.2, S. 382–398.

Gendreau, Michel und Potvin, Jean-Yves (2005). „Metaheuristics in Combinatorial Optimization“. In: *Annals of Operations Research* 140.1, S. 189–213.

Gerdes, I., Klawonn, F. und Kruse, R. (2004). *Evolutionäre Algorithmen:* Computational Intelligence. Vieweg+Teubner Verlag.

Günther, H.-O. und Tempelmeier, H. (2012). *Produktion und Logistik*. Springer-Verlag, Berlin.

Gomes, Carla P u. a. (2000). „Heavy-tailed phenomena in satisfiability and constraint satisfaction problems“. In: *Journal of automated reasoning* 24.1-2, S. 67–100.

Gorry, G. A. und Morton, Michael S. (1971). *A framework for management information systems*. Cambridge, M.I.T.

Görz, Günther, Rollinger, Claus-Rainer und Schneeberger, Josef, Hrsg. (2003). *Handbuch der Künstlichen Intelligenz*. 4. Aufl. München: Oldenbourg.

Gottlob, Georg, Leone, Nicola und Scarcello, Francesco (2000). „A comparison of structural CSP decomposition methods“. In: *Artificial Intelligence* 124.2, S. 243–282.

Guesgen, Hans W. (2003). „Constraints“. In: *Handbuch der Künstlichen Intelligenz*. Hrsg. von Günther Görz, Claus-Rainer Rollinger und Josef Schneeberger. 4. Aufl. München: Oldenbourg.

Hadzic, Tarik und Andersen, Henrik Reif (2004). „An introduction to solving interactive configuration problems“. In: *Techn. Ber. TR-2004-49, The IT University of Copenhagen*.

Hadzic, Tarik u. a. (2004). „Fast Backtrack-Free Product Configuration using a Precompiled Solution Space Representation“. In: *Proceedings of the International Conference on Economic, Technical and Organisational Aspects of Product Configuration Systems*, S. 131–138.

Hartmann, Sönke und Briskorn, Dirk (2010). „A survey of variants and extensions of the resource-constrained project scheduling problem“. In: *European Journal of Operational Research* 207.1, S. 1–14.

Höckner, Benny (2011). „Stundenplanung mit Choco“. Studienarbeit. Brandenburgische Technische Universität Cottbus.

Heisig, G. (2002). *Planning Stability in Material Requirements Planning Systems*. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg.

Hengl, Hans-Thomas (2011). „Mobile ERP-Projekte: Anwender in der Warteschleife“. In: *ZDNet [`http://www.zdnet.de/magazin/41558620/mobile-erp-projekte-anwender-in-der-warteschleife.htm`, abgerufen am 9.7.2017]*.

Higgins, Peter G. (1998). „Extending Cognitive Work Analysis to Manufacturing Scheduling“. In: *Proceedings of the Australasian Conference on Computer Human Interaction*. OZCHI '98. Washington, DC, USA: IEEE Computer Society, S. 236–243.

– (1999). „Job shop scheduling: Hybrid intelligent human-computer paradigm“. Dissertation. University of Melbourne, Australia.

– (2001). „Architecture and interface aspects of scheduling decision support“. In: *Human performance in planning and scheduling*. Taylor & Francis, S. 245–280.

Hillier, F.S. und Lieberman, G.J. (2001). *Introduction to Operations Research*. McGraw-Hill International Editions. McGraw-Hill.

Hoc, Jean-Michel (2000). „From human – machine interaction to human – machine cooperation“. In: *Ergonomics* 43.7, S. 833–843.

Hoc, Jean-Michel, Guerin, Clément und Mebarki, Nasser (2012). „The nature of expertise in scheduling: The case of timetabling“. In: *Human Factors and Ergonomics in Manufacturing and Service Industries*.

Hofstedt, P. und Wolf, A. (2007). *Einführung in die Constraint-Programmierung: Grundlagen, Methoden, Sprachen, Anwendungen*. Springer.

Hong, Lianxi (2012). „An improved {LNS} algorithm for real-time vehicle routing problem with time windows“. In: *Computers & Operations Research* 39.2, S. 151–163.

Huber, M. (2011). *Verfahren zur Grobplanung bei Einzel-, Serien-, und Massenfertigung sowie die Integration in PPS-Systemen*. GRIN Verlag.

IFS (2011). *IFS Industry Reports and Studies: IFS Usability Survey Report - Does ERP Mean Excel Runs Production?* IFS North America.

Igbaria, Magid u. a. (1996). „The impact and benefits of a DSS: The case of Fleet-Manager“. In: *Information & Management* 31.4, S. 215–225.

Ingimundardottir, Helga und Runarsson, ThomasPhilip (2011). „Supervised Learning Linear Priority Dispatch Rules for Job-Shop Scheduling“. In: *Learning and Intelligent Optimization*. Hrsg. von CarlosA.Coello Coello. Bd. 6683. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 263–277.

Jackson, Sarah, Wilson, John R. und MacCarthy, Bart L. (2004). „A New Model of Scheduling in Manufacturing: Tasks, Roles, and Monitoring“. In: *Human Factors* 46.3, S. 533–550.

Jaâfar, Inès Ben, Khayati, Naoufel und Ghédira, Khaled (2004). „Multicriteria Optimization in CSPs : Foundations and Distributed Solving Approach.“ In: *AIMSA*. Hrsg. von Christoph Bussler und Dieter Fensel. Bd. 3192. Lecture Notes in Computer Science. Springer, S. 459–468.

Jones, Randolph M. und Langley, Pat (2005). „A Constrained Architecture for Learning and Problem Solving“. In: *Computational Intelligence* 21.4, S. 480–502.

Jussien, Narendra und Barichard, Vincent (2000). „The PaLM system: explanation-based constraint programming“. In: *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP*. Bd. 2000, S. 118–133.

Kallehauge, Brian u. a. (2005). „Vehicle Routing Problem with Time Windows“. In: *Column Generation*. Hrsg. von Guy Desaulniers, Jacques Desrosiers und MariusM. Solomon. Springer US, S. 67–98.

Kasprik, R. (2002). *Rationale Unternehmens- Und Marketingplanung: Strategische, Operative Und Taktische Entscheidungen*. Physica-Verlag.

Kazakci, A. O. und Tsoukiàs, A. (2003). *Designing or Planning? - Cognitive foundations for design aiding*. Cahier du Lamsade No 214. Université Paris Dauphine.

Kiewitt, Anja (2010). „Ein Drittel ohne ERP“. In: *Logistik Heute [http://www.logistik-heute.de/Logistik-News-Logistik-Nachrichten/Markt-*

News/8047/Umfrage-KMU-wollen-Spezial-Loesungen-aus-einer-Hand-Ein-Drittel-ohne-ERP, abgerufen am 9.7.2017].

King, Rachael (2012). „SAP Owns Up to Usability Problem“. In: *CIO Journal [http://blogs.wsj.com/cio/2012/08/02/sap-owns-up-to-usability-problem/, abgerufen am 9.7.2017].*

Klau, Gunnar W. u. a. (2010). „Human-guided search“. In: *Journal of Heuristics* 16.3, S. 289–310.

Klein, Michel, Lecomte, Catherine und Dejax, Pierre (1993). „Using a knowledge-based decision support system development environment to implement a scheduling system for a workshop“. In: *International Journal of Production Economics* 30-31, S. 437–451.

Klein, Michel und Methlie, Leif B. (2009). *Knowledge-Based Decision Support Systems With Applications in Business: A Decision Support Approach*. 2nd. John Wiley & Sons.

Konradin (2011). *Konradin ERP-Studie 2011: Einsatz von ERP-Lösungen in der Industrie*. Konradin Mediengruppe.

Kopfer, Herbert und Schonberger, J (2002). „Interactive solving of vehicle routing and scheduling problems: Basic concepts and qualification of tabu search approaches“. In: *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*. IEEE, S. 1425–1434.

Korte, B., Vygen, J. und Randow, R. von (2012). *Kombinatorische Optimierung: Theorie und Algorithmen*. Springer-Lehrbuch Masterclass. Springer.

Kovács, András (2005). „Novel Models and Algorithms for Integrated Production Planning and Scheduling“. Dissertation. Department of Measurement u. a.

Kravchenko, Svetlana A. (2000). „Minimizing the number of late jobs for the two-machine unit-time job-shop scheduling problem“. In: *Discrete Applied Mathematics* 98.3, S. 209–217.

Kurbel, KarlE. (2013). „MRP: Material Requirements Planning“. In: *Enterprise Resource Planning and Supply Chain Management*. Progress in IS. Springer Berlin Heidelberg, S. 19–60.

Kuster, Jürgen, Jannach, Dietmar und Friedrich, Gerhard (2007). „Handling Alternative Activities in Resource-constrained Project Scheduling Problems“. In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI'07. Morgan Kaufmann Publishers Inc., S. 1960–1965.

Lambeck, Christian u. a. (2012a). „Bridging the gap: advances in interaction design for enterprise applications in production scenarios“. In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. AVI '12. ACM, S. 765–768.

Lambeck, Christian u. a. (2012b). „Changing Concepts in Human-Computer-Interaction in Real-time Enterprise Systems - Introducing a Concept for Intuitive Decision Support in SCM Scenarios“. In: *ICEIS (1)*. Hrsg. von Leszek A. Maciaszek, Alfredo Cuzzocrea und José Cordeiro. SciTePress, S. 139–144.

Langley, Pat und Rogers, Seth (1958). „An Extended Theory of Human Problem Solving“. In: *International Journal of Solar System Studies* November 2007, S. 1242–1247.

Laporte, Gilbert u. a. (2000). „Classical and modern heuristics for the vehicle routing problem“. In: *International transactions in operational research* 7.4-5, S. 285–300.

Lawler, Eugene L u. a. (1993). „Sequencing and scheduling: Algorithms and complexity“. In: *Handbooks in operations research and management science* 4, S. 445–522.

Lecoutre, C. (2010). *Constraint Networks: Targeting Simplicity for Techniques and Algorithms*. Wiley.

– (2013). *Constraint Networks: Targeting Simplicity for Techniques and Algorithms*. ISTE. Wiley.

Lei, Hongtao, Laporte, Gilbert und Guo, Bo (2011). „The capacitated vehicle routing problem with stochastic demands and time windows“. In: *Computers & Operations Research* 38.12, S. 1775–1783.

Lesh, Neal u. a. (2003). „Human-Guided Search for Jobshop Scheduling“. In: *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*. Houston, Texas.

Leyh, Christian und Gottwald, Henrique (2011). „Nutzung von ERP-Systemen in deutschen klein- und mittelständischen Unternehmen“. In: *Dresdner Beiträge zur Wirtschaftsinformatik*.

Liouane, Noureddine u. a. (2007). „Ant systems & local search optimization for flexible job shop scheduling production“. In: *International Journal of Computers, Communications & Control* 2.2, S. 174–184.

Luz, Saturnino und Masoodian, Masood (2010). „Improving focus and context awareness in interactive visualization of time lines“. In: *Proceedings of the 24th BCS Interaction Specialist Group Conference*. BCS '10. British Computer Society, S. 72–80.

MacCarthy, Bart (2006). „Organizational, Systems and Human Issues in Production Planning, Scheduling and Control“. In: *Handbook of Production Scheduling*. Hrsg. von JeffreyW. Herrmann. Bd. 89. International Series in Operations Research & Management Science. Springer US, S. 59–90.

MacCarthy, B.L., Wilson, J.R. und Crawford, S. (2001). „Human performance in industrial scheduling: A framework for understanding“. In: *Human Factors and Ergonomics in Manufacturing and Service Industries* 11.4, S. 299–320.

MacDonald, M. (2008). *Your Brain: The Missing Manual*. Missing manual. O'Reilly Media.

Madsen, Jeppe Nejsum (2003). „Methods for interactive constraint satisfaction“. Master's Thesis. Department of Computer Science, University of Copenhagen.

Marcus, Aaron (2005). „The ROI of Usability“. In: *Cost-Justifying Usability: An Update for the Internet Age, Second Edition*. Hrsg. von R.G. Bias und D.J. Mayhew. Morgan Kaufmann Series in Interactive Technologies. Elsevier Science.

Markus, Chimani u. a. (2005). „A Case Study in Large-Scale Interactive Optimization“. In: *Artificial Intelligence and Applications* 453, S. 192.

Matthews, Dan (2008). *IFS Research Report: The Increasing Importance Of Usability in Enterprise Software*. IFS North America.

McCollum, Barry (2006). „University Timetabling: Bridging the Gap between Research and Practice“. In: *in Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*. Springer, S. 15–35.

– (2007). „A Perspective on Bridging the Gap Between Theory and Practice in University Timetabling“. In: *Practice and Theory of Automated Timetabling VI*. Hrsg. von EdmundK. Burke und Hana Rudová. Bd. 3867. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 3–23.

McIlroy, Rich C. und Stanton, Neville A. (2011). „Getting past first base: Going all the way with Cognitive Work Analysis“. In: *Applied Ergonomics* 42.2, S. 358–370.

McKay, Kenneth N. und Buzacott, John A. (2000). „The application of computerized production control systems in job shop environments“. In: *Computers in Industry* 42.23, S. 79–97.

McKay, Kenneth N. und Wiers, Vincent C.S. (1999). „Unifying the theory and practice of production scheduling“. In: *Journal of Manufacturing Systems* 18.4, S. 241–255.

– (2002). „Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory“. In: *Computers in Industry* 50, S. 5–14.

Mietus, D.M. (1994). „Understanding Planning for Effective Decision Support: A Cognitive Task Analysis of Nurse Scheduling“. Dissertation. University of Groningen, Netherlands.

Müller, Tomáš (2003). „Interactivity in Constraint Programming“. In: *Principles and Practice of Constraint Programming – CP 2003*. Hrsg. von Francesca Rossi. Bd. 2833. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 986–986.

Müller, Tomáš und Barták, Roman (2002). „Interactive Timetabling: Concepts, Techniques, and Practical Results“. In: *PATAT 2002 — Proceedings of the 4th international conference on the Practice And Theory of Automated Timetabling*, S. 58–72.

Morikawa, Katsumi und Takahashi, Katsuhiko (2006). „Modeling Planning and Scheduling Tasks Based on Interviews“. In: *Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference 2006*.

Morris, Michael G. und Dillon, Andrew (1997). „The Influence of User Perceptions on Software Utilization: Application and Evaluation of a Theoretical Model of Technology Acceptance“. In: *IEEE SOFTWARE* 14, S. 58–76.

Nascimento, Hugo A.D. do und Eades, Peter (2005). „User hints: a framework for interactive optimization“. In: *Future Generation Computer Systems* 21.7, S. 1177–1191.

Nie, Li u. a. (2013). „Reactive scheduling in a job shop where jobs arrive over time“. In: *Computers & Industrial Engineering* 66.2, S. 389–405.

Nielsen, J. (1994). *Usability Engineering*. Interactive Technologies. Elsevier Science.

Nielsen, Jakob und Giluz, Shuli: (2007). *Usability Return on Investment*. Fremont: Nielsen Norman Group.

Nittler, Mark (2007). „IT evolution: Why ERP systems face extinction“. In: *W.P. Carey KnowIT[https://blogs.wpcarey.asu.edu/knowit/it-evolution-why-erp-systems-face-extinction/, abgerufen am 9.7.2017].*

Noronha, SJ und Sarma, VVS (1991). „Knowledge-based approaches for scheduling problems: A survey“. In: *Knowledge and Data Engineering, IEEE Transactions on* 3.2, S. 160–171.

O'Callaghan, Barry, O'Sullivan, Barry und Freuder, EugeneC. (2005). „Generating Corrective Explanations for Interactive Constraint Satisfaction“. In: *Principles and Practice of Constraint Programming - CP 2005*. Hrsg. von Peter van Beek. Bd. 3709. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 445–459.

Ochoa, G. u. a. (2009). „Dispatching rules for production scheduling: A hyper-heuristic landscape analysis“. In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*, S. 1873–1880.

Oezbayrak, Mustafa und Bell, Robert (2003). „A knowledge-based decision support system for the management of parts and tools in FMS“. In: *Decision Support Systems* 35.4, S. 487–515.

Oja, M-K. und Lucas, W. (2010). „Evaluating the Usability of ERP Systems: What Can Critical Incidents Tell Us?“ In: *Pre-ICIS Workshop on Enterprise Systems Research in MIS.*

Ombuki, Beatrice M und Ventresca, Mario (2004). „Local search genetic algorithms for the job shop scheduling problem“. In: *Applied Intelligence* 21.1, S. 99–109.

Osintsev, Aleksey (2012). „Usability Still a Problem for ERP Users“. In: *Technology Evaluation Centers Blog [http://blog.technologyevaluation.com/blog/2012/11/20/usability-issues-in-erp-systems/, abgerufen am 9.7.2017].*

Panwalkar, S. S. und Iskander, Wafik (1977). „A Survey of Scheduling Rules“. In: *Operations Research* 25.1, S. 45–61.

Parasuraman, R., Sheridan, T. B. und Wickens, C. D. (2000). „A model for types and levels of human interaction with automation“. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* 30.3, S. 286–297.

Parasuraman, Raja und Manzey, Dietrich H. (2010). „Complacency and Bias in Human Use of Automation: An Attentional Integration“. In: *Human Factors* 52.3, S. 381–410.

Pillac, Victor u. a. (2013). „A review of dynamic vehicle routing problems“. In: *European Journal of Operational Research* 225.1, S. 1–11.

Pinedo, Michael L. (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer New-York.

Pochet, Yves (2001). „Mathematical Programming Models and Formulations for Deterministic Production Planning Problems“. In: *Computational Combinatorial Optimization*. Hrsg. von Michael Jünger und Denis Naddef. Bd. 2241. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 57–111.

Portougal, Victor und Robb, David J. (2000). „Production Scheduling Theory: Just Where Is It Applicable?“ In: *Interfaces* 30.6, S. 64–76.

Prenzel, Anna (2011). „Entwurf eines interaktiven Entscheidungsunterstützungssystems zur Routenplanung“. Masterarbeit. Hochschule Zittau/Görlitz.

Prenzel, Anna und Ringwelski, Georg (2011). „Description of a Practical, Benders' Cut Inspired VRP System“. In: *Proceedings International Workshop on Innovative Scheduling and other Applications using CP-AI-OR (ISA 2011). Computer Science Report 01/11 of Brandenburg University of Technology Cottbus*.

– (2012a). „Design of Human-computer Interfaces in Scheduling Applications“. In: *ICEIS (1)*. Hrsg. von Leszek A. Maciaszek, Alfredo Cuzzocrea und José Cordeiro. SciTePress, S. 219–228.

Prenzel, Anna und Ringwelski, Georg (2012b). „Statistical Evaluation of the Usability of Decision-Oriented Graphical Interfaces in Scheduling Applications“. In: *GI-Jahrestagung*. Hrsg. von Ursula Goltz u. a. Bd. 208. LNI. GI, S. 225–236.

Pritchett, Amy R, Kim, So Young und Feigh, Karen M (2014). „Modeling Human–Automation Function Allocation“. In: *Journal of Cognitive Engineering and Decision Making* 8.1, S. 33–51.

Prot, D., Bellenguez-Morineau, O. und Lahlou, C. (2013). „New complexity results for parallel identical machine scheduling problems with preemption, release dates and regular criteria“. In: *European Journal of Operational Research* 231.2, S. 282–287.

Pu, Pearl und Faltings, Boi (2002). „Effective Interaction Principles for User-Involved Constraint Problem Solving“. In: *Second International Workshop on User-Interaction in Constraint Satisfaction, the Eighth International Conference on Principles and Practice of Constraint Programming*.

– (2004). „Decision tradeoff using example-critiquing and constraint programming“. In: *Constraints* 9.4, S. 289–310.

Qu, Rong und He, Fang (2009). „A Hybrid Constraint Programming Approach for Nurse Rostering Problems“. In: *Applications and Innovations in Intelligent Systems XVI*. Hrsg. von Tony Allen, Richard Ellis und Miltos Petridis. Springer London, S. 211–224.

Rahimi, Mansour und Dessouky, Maged (2001). „A hierarchical task model for dispatching in computer-assisted demand-responsive paratransit operation“. In: *Journal of Intelligent Transportation Systems* 6.3, S. 199–223.

Rasmussen, J. (1983). „Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models“. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.3, S. 257–266.

Régin, Jean-Charles (2004). „Global constraints and filtering algorithms“. In: *Constraint and Integer Programming*. Springer, S. 89–135.

Rettig, Cynthia (2007). „The Trouble With Enterprise Software“. In: *MITSloan Managment Review* 49.1.

Riedel, Ralph u. a. (2011). „Building Decision Support Systems for Acceptance“. In: *Behavioral Operations in Planning and Scheduling*. Hrsg. von Jan C. Fransoo, Toni Waefler und John R. Wilson. Springer Berlin Heidelberg, S. 231–295.

Rossi, Francesca, van Beek, Peter und Walsh, Toby (2006). *Handbook of constraint programming*. Elsevier.

Rubin, Jeffrey (1994). *Handbook of usability testing: how to plan, design, and conduct effective tests*. Wiley technical communication library. Wiley.

Ruiz, Rubén, Maroto, Concepción und Alcaraz, Javier (2004). „A decision support system for a real vehicle routing problem“. In: *European Journal of Operational Research* 153.3, S. 593–606.

Russell, Stuart J. und Norvig, Peter (2012). *Künstliche Intelligenz. Ein moderner Ansatz*. 3., überarb. A. Pearson Studium.

Ryan, Vincent (2009). „ERP Made Easy?“ In: *CFO Magazine*.

Sanderson, Penelope M. (1989). „The Human Planning and Scheduling Role in Advanced Manufacturing Systems: An Emerging Human Factors Domain“. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 31.6, S. 635–666.

Santa-Eulalia, Luis Antonio de u. a. (2011). „Advanced Supply Chain Planning Systems (APS) Today and Tomorrow“. In: *Supply Chain Management - Pathways for Research and Practice*. Hrsg. von Prof. Dilek Onkal. InTech.

Saygin, C. und Kilic, S. E. (1999). „Integrating Flexible Process Plans with Scheduling in Flexible Manufacturing Systems“. In: *The International Journal of Advanced Manufacturing Technology* 15.4, S. 268–280.

Schneeweiss, Denny und Hofstedt, Petra (2013). „FdConfig: A Constraint-Based Interactive Product Configurator“. In: *Applications of Declarative Programming and Knowledge Management*. Hrsg. von Hans Tompits u. a. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 239–255.

Schneider, J. und Kirkpatrick, S. (2006). *Stochastic Optimization*. Scientific Computation. Springer Berlin Heidelberg, S. 59–61.

Schneider, Wolfgang (2008). *Ergonomische Gestaltung von Benutzungsschnittstellen: Kommentar zur Grundsatznorm DIN EN ISO 9241-110*. Verlag Beuth.

Sels, Veronique, Gheysen, Nele und Vanhoucke, Mario (2012). „A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions“. In: *International Journal of Production Research* 50.15, S. 4255–4270.

Shih, Ya-Yueh und Huang, Siao-Sian (2009). „The Actual Usage of ERP Systems: An Extended Technology Acceptance Perspective.“ In: *Journal of Research and Practice in Information Technology* 41.3, S. 263–276.

Smith, Stephen F., Hildum, David W. und Crimm, David R. (2005). „Comirem: An Intelligent Form for Resource Management“. In: *IEEE Intelligent Systems* 20.

Snoo, Cees de, Wezel, Wout van und Jorna, René J. (2011). „An empirical investigation of scheduling performance criteria“. In: *Journal of Operations Management* 29.3, S. 181–193.

Snoo, Cees. de u. a. (2011). „Coordination activities of human planners during rescheduling: case analysis and event handling procedure“. In: *International Journal of Production Research* 49.7, S. 2101–2122.

Solomon, Marius M (1987). „Algorithms for the vehicle routing and scheduling problems with time window constraints“. In: *Operations research* 35.2, S. 254–265.

Sontow, Dr. Karsten, Treutlein, Peter und Sontow, Rainer (2014). *ERP in der Praxis - Anwenderzufriedenheit, Nutzen & Perspektiven*. TROVARIT AG.

Stacey, Martin und Eckert, Claudia (2010). „Reshaping the box: creative designing as constraint management“. In: *International Journal of Product Development* 11.3/4, S. 241–255.

Staedler, Hartmut u. a. (2012). *Advanced Planning in Supply Chains - Illustrating the Concepts Using an SAP APO Case Study*. Springer Berlin Heidelberg.

Stenzel, Martin Peter (2012). „Eine interaktive GUI für die constraint-basierte Stundenplanung“. Bachelorarbeit. Brandenburgische Technische Universität Cottbus.

Stoop, Paul P. M. und Wiers, Vincent C.S. (1996). „The complexity of scheduling in practice“. In: *International Journal of Operations & Production Management* 16 (10), S. 37–53.

Stormer, Henrik (2007). „Kundenbasierte Produktkonfiguration“. In: *Informatik-Spektrum* 30.5, S. 322–326.

Timpe, Christian (2002). „Solving planning and scheduling problems with combined integer and constraint programming“. In: *OR Spectrum* 24.4, S. 431–448.

Tompkins, J.A. (2010). *Facilities Planning*. John Wiley & Sons.

Topi, Heikki, Lucas, Wendy und Babaian, Tamara (2005). „Identifying usability issues with an ERP implementation“. In: *Proceedings of the International Conference on Enterprise Information Systems ICEIS-2005*, S. 128–133.

Toth, P. und Vigo, D. (2002). *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial und Applied Mathematics.

Tullis, Tom und Albert, Bill (2008). *Measuring the User Experience*. Morgan Kaufmann Series in Interactive Technologies. Elsevier Science.

Turban, Efraim, Sharda, Ramesh und Delen, Dursun (2011). *Decision Support and Business Intelligence Systems*. Pearson.

Usher, John M. und Kaber, David B. (2000). „Establishing information requirements for supervisory controllers in a flexible manufacturing system using GTA“. In: *Human Factors and Ergonomics In Manufacturing* 10.4, S. 431–452.

van Beek, Peter (2006). „Backtracking Search Algorithms“. In: *Handbook of Constraint Programming*. Hrsg. von Francesca Rossi, Peter van Beek und Toby Walsh. Elsevier. Kap. 4.

Van Hoesel, Stan u. a. (2002). *Polynomial time algorithms for some multi-level lot-sizing problems with production capacities*. Techn. Ber. Tinbergen Institute Discussion Paper.

Vazirani, V.V. (2001). *Approximation Algorithms*. Springer.

Vicente, Kim J. (1999). *Cognitive work analysis: Towards safe, productive, and healthy computer-based work*. Mahwah, NJ: Lawrence Erlbaum.

Vicente, Kim J. und Rasmussen, J. (1992). „Ecological interface design: theoretical foundations“. In: *IEEE Transactions on Systems, Man, and Cybernetics* 22.4, S. 589–606.

Vieira, Guilherme E., Herrmann, Jeffrey W. und Lin, Edward (2003). „Rescheduling manufacturing systems: A framework of strategies, policies, and methods“. In: *Journal of Scheduling* 6, S. 39–62.

Wezel, Wout van (2006). „Interactive Scheduling Systems“. In: *Planning in Intelligent Systems*. Hrsg. von Wout van Wezel, René J. Jorna und Alexander M. Meystel. Wiley-Interscience, S. 205–243.

Wäfler, Toni (2001). „Planning and scheduling in secondary work systems“. In: *Human performance in planning and scheduling*. Taylor & Francis, S. 411–449.

Wäfler, Toni u. a. (2011). „Human Control Capabilities“. In: *Behavioral Operations in Planning and Scheduling*. Hrsg. von Jan C. Fransoo, Toni Waefler und John R. Wilson. Springer Berlin Heidelberg, S. 47–81.

Wiers, Vincent C. S. (1997a). „A review of the applicability of OR and AI scheduling techniques in practice“. In: *Omega* 25.2, S. 145–153.

– (1997b). „Decision support systems for production scheduling tasks- Part I of a case study: Analysis and task redesign“. In: *Production Planning & Control* 8.7, S. 711–721.

Wiers, Vincent C. S. und Van Der Schaaf, Tjerk W. (1997). „A framework for decision support in production scheduling tasks“. In: *Production Planning & Control* 8.6, S. 533–544.

Wierzbicki, A.P., Makowski, M. und Wessels, J., Hrsg. (2000). *Model-Based Decision Support Methodology with Environmental Applications*. Series: Mathematical Modeling and Applications. Dordrecht: Kluwer Academic.

Woeginger, GerhardJ. (2003). „Exact Algorithms for NP-Hard Problems: A Survey“. In: *Combinatorial Optimization — Eureka, You Shrink!* Hrsg. von Michael Jünger, Gerhard Reinelt und Giovanni Rinaldi. Bd. 2570. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 185–207.

Wong, B. L. William und Bl, Ann (2002). „Analysing ambulance dispatcher decision making: Trialing Emergent Themes Analysis. Paper presented at the Human“. In: *Factors 2002, the Joint Conference of the Computer Human Interaction Special Interest Group and The Ergonomics Society of Australia, HF2002*.

Woywode, Prof. Dr. Michael u. a. (2012). „Abschlussbericht: Gebrauchstauglichkeit von Anwendungssoftware als Wettbewerbsfaktor für kleine und mittlere Unternehmen (KMU)“. In: *Usability in Germany [`http://www.usability-in-germany.de/ergebnis`, abgerufen am 9.7.2017]*.

Xiong, Jian, Xing, Li ning und Chen, Ying wu (2013). „Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns“. In: *International Journal of Production Economics* 141.1, S. 112–126.

Zandieh, M. und Adibi, M.A. (2010). „Dynamic job shop scheduling using variable neighbourhood search“. In: *International Journal of Production Research* 48.8, S. 2449–2458.

Index

Anhang A

Usability-Testplan

Dieser Abschnitt enthält den vollständigen Testplan, der von mir für die Vorbereitung des Usability-Tests nach den Empfehlungen von (Rubin, 1994) erstellt wurde. Die Zuordnung von Testgruppen zu Konfigurationen und die Testaufgaben beziehen sich auf den zweiten Testdurchgang. Die Vorbereitung des ersten Durchgangs erfolgte analog (vgl. Prenzel und Ringwelski (2012a) und Prenzel und Ringwelski (2012b)).

COMORES

Entscheidungsunterstützungssystem für die Flottenplanung

Usability-Testplan

Hochschule Zittau/Görlitz
Fachbereich Informatik

1 Testziel

Entscheidungsunterstützungssysteme (EUS) erleichtern dem Anwender durch vielfältige Interaktionsmöglichkeiten die Lösung komplexer Fragestellungen. Zu den Einsatzgebieten gehören Dispositionsprobleme, bei denen die Zuordnung von Aufträgen (Jobs) zu Ressourcen im Vordergrund steht. Sie treten bei der Routenplanung für Fahrzeugflotten oder in der Produktionsplanung auf.

Die manuelle Verplanung von zahlreichen Aufträgen ist in der Regel sehr mühsam und zeitaufwändig. Es müssen zahlreiche Randbedingungen und Abhängigkeiten der Aufträge und Ressourcen beachtet werden. Aus diesem Grund wurden Softwarewerkzeuge entwickelt, die eine automatische Planung ermöglichen. Diese berücksichtigen jedoch oft nicht die individuellen Planungswünsche der Disponenten. Daher lehnen sie diese Werkzeuge häufig ab und arbeiten weiterhin manuell.

Wir forschen daran, wie man Planungssysteme so gestalten kann, dass manuelle Planänderungen bequem vorgenommen werden können. Ziel ist es, dabei die Vorteile der automatischen Planung weitestgehend zu erhalten.

Unser Anwendungsbereich ist dabei die Planung von Fahrzeugflotten. Damit ergeben sich folgende Interpretationen für *Jobs* und *Ressourcen:*

- Jobs sind Aufträge, die an bestimmten Orten zu erledigen sind. Sie werden pro Fahrzeug zu einer Tour angeordnet. Da zwischen den einzelnen Orten Fahrstrecken zurückgelegt werden müssen, hängen die Kosten einer Tour wesentlich von der Reihenfolge der Jobs ab.
- Ressourcen sind Fahrzeuge bzw. Personen, welche die Touren abarbeiten.

Jobs und Ressourcen lassen sich durch verschiedene Eigenschaften spezifizieren.

Jobs:

Zeitfenster	Regelt den frühest- und spätestmöglichen Beginn eines Jobs.
Dauer	Die Ausführungszeit am Ort in Minuten.
Qualifikation	Die Qualifikation muss mit der Qual. der Ressource übereinstimmen. Wenn keine angegeben ist, kann jede Ressource eingesetzt werden.
Jobtyp	Normal oder Pause
Ort	Ausführungsort
Ressource	Fahrzeug, welches den Job übernimmt. Die Ressource wird - fest voreingestellt oder - während der manuellen Planung oder - der automatischen Optimierung ermittelt.
Startzeit	Die Startzeit wird - wird fest voreingestellt (Zeitfenster) oder - während der manuellen Planung oder - der automatischen Optimierung ermittelt.

Ressourcen:

Qualifikation	Regelt, welcher Job übernommen werden kann. Ist keine angegeben, kann jeder Job übernommen werden, ansonsten nur Jobs mit gleicher oder keiner Qualifikation.

Die Erzeugung von Plänen kann nach zwei Optimierungskriterien erfolgen: *minimale Fahrzeit für alle Ressourcen* oder *Gleichverteilung der Jobs auf die Ressourcen.*

Der Prototyp unseres Flottenplanungssystems gehorcht der obigen Spezifikation von Jobs und Ressourcen und enthält folgende Interaktionselemente:

- *Vollständige automatische Planung*
  - Optimierung aller eingegebenen Jobs

- *Automatische Planung von Jobgruppen*
  - ressourcenübergreifend im Plan oder in der Ablage
  - Optimierungseinstellungen: Reihenfolge *oder* Startzeit von restlichen Jobs im Plan fixieren

- *Automatische Planung von Ressourcen*
  - Optimierung der Reihenfolge innerhalb einer Ressource

- *Konfliktauflösung*
  - Aufheben von Jobüberlappungen innerhalb einer Ressource unter Beibehaltung der Reihenfolge (auch als „Einordnen“ bezeichnet)

- *Fixierung von Jobs und Jobgruppen*
  - Gleichzeitige Fixierung von Startzeit und Ressource im Plan
  - Die Fixierung bleibt während einer automatischen Optimierung erhalten

- *Suchraumvisualisierung (Enhanced Constraint Highlighting)*
  - Farbliche Hinterlegung von Zeitfenster und Qualifikation im Plan
  - Zusätzlich Algorithmische Reduzierung des Suchraums, um gültige Positionen für Jobs im Plan hervorzuheben

- *Dynamische Informationsanzeige*
  - Fahrzeitanzeige während Drag and Drop
  - Qualitätsanzeige während Drag and Drop

Mit diesen Features soll eine optimale Mensch- Computer-Interaktion zur Erstellung praxistauglicher Pläne ermöglicht werden. Der Nachweis dafür soll mit Hilfe der durchzuführenden Usability-Tests erbracht werden. Die Testpersonen arbeiten mit verschiedenen Konfigurationen des Prototyps. Je nach Konfiguration sind einige Features aktiviert oder deaktiviert. Es wird erwartet, dass mit aktiviertem Feature jeweils bessere Pläne erzeugt werden.

Eine Analyse des Anwendungsbereiches hat ergeben, dass sich Planungsentscheidungen des Disponenten aus

- der Vorgabe von Startzeiten für Jobs
- der Vorgabe von Ressourcen für Jobs
- der Vorgabe einer Reihenfolge für bestimmte Jobs
- der Festlegung eines Optimierungskriteriums für Jobgruppen

zusammensetzen. Diese Aktionen lassen sich mit dem EUS-Prototyp abbilden.

Als Testpersonen werden keine Disponenten, sondern Studenten und andere freiwillige Teilnehmer ohne Expertenwissen eingesetzt. Dies ist möglich, da in den Tests das Expertenwissen durch Vorgabe der oben angegebenen Aktionen simuliert wird. Beispielsweise kann der Testperson die Verwendung einer festen Startzeit für einen Job vorgegeben werden. Ein Anwendungsexperte würde die Notwendigkeit dieser Einschränkung aus seinem Hintergrundwissen beziehen. Trotzdem ergibt sich dieselbe Umsetzung mit Hilfe der Interaktionselemente.

2 Durchführung und Betreuung der Testläufe

1.1 Allgemeines

Das System befindet sich unter der URL http://comores.inf.hs-zigr.de/eus_flottenplanung

2.1 Gruppierung der Teilnehmer

An den Usability-Tests können beliebige Personen teilnehmen. Lediglich der grundlegende Umgang mit dem Computer und die Bedienung grafischer Benutzeroberflächen mit der Maus sollten den Teilnehmern vertraut sein. Das weitere benötigte Wissen wird vor der Durchführung der Tests vermittelt.

Um die Leistungsfähigkeit des Entwurfs unseres EUS zu messen, wurden die in Bild 1 gezeigten Konfigurationen **M1 – M5** erstellt. In jeder Konfiguration sind bestimmte Features aktiviert bzw. deaktiviert.

	M1	M2	M3	M4	M5
Enhanced Constraint Highlighting	●	●	●	●	●
Konfliktauflösen				●	●
Gruppenplanung				●	●
Ressourcenplanung				●	●
Volloptimierung		●	●		●
Fixierung			●	●	●

Bild 1: Aktivierung der Features in verschiedenen Konfigurationen

Pro Konfiguration wird eine Testgruppe benötigt, die aus mindestens 7 Personen bestehen muss. Jeder Test muss in allen Konfigurationen durchlaufen werden. In Bild 2 ist dargestellt, in welcher Konfiguration jede Testgruppe die jeweiligen Tests durchführt. Es wurde darauf Wert gelegt, dass

- Jeder Test in jeder Konfiguration durchgeführt wird
- Jeder Testgruppe pro Test eine andere Konfiguration zugeteilt wird.

Daraus ergibt sich die “Sudoku”-Aufteilung. Sie ermöglicht verschiedene Auswertungskriterien:

- Vergleich der Konfigurationen pro Test (Wie gut sind bestimmte Konfigurationen und/oder Features für bestimmte Aufgabenstellungen geeignet?)
- Vergleich der Konfigurationen pro Testgruppe
- Befragung der Gruppenteilnehmer zu Vor- und Nachteilen bestimmter Konfigurationen.

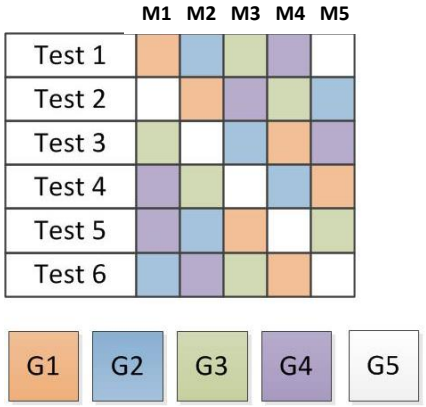


Bild 2: Zuordnung von Testgruppen zu Tests in bestimmten Konfigurationen

Bei mind. 7 Testpersonen pro Gruppe werden folglich mind. 35 Testpersonen benötigt.

Da die Konfigurationen aufeinander aufbauen, empfiehlt es sich, pro Gruppe die Tests nach den Konfigurationen zu sortieren.

Gruppe	Testreihenfolge
G1	Test1, Test2, Test5, Test3, Test6, Test4
G2	Test6, Test1, Test5, Test3, Test4, Test2
G3	Test3, Test4, Test1, Test6, Test2, Test5
G4	Test4, Test5, Test6, Test2, Test1, Test3
G5	Test2, Test3, Test4, Test5, Test6, Test1

2.2 Training der Teilnehmer

Die Einweisung setzt sich wie folgt zusammen:

(1) Kurze Zusammenfassung der Zielstellung aus Abschnitt 1. Es geht darum, Pläne so zu modifizieren, dass spezielle Wünsche des Disponenten berücksichtigt werden. Trotz der Modifikationen soll die Gesamtfahrzeit möglichst optimal bleiben.

(2) Erklärung der Spezifikation für
   (a) Jobs
   (b) Ressourcen
   (c) Optimierungsziele

(3) Erläuterung der Optimierungsfeatures anhand einer Live-Demonstration des Systems. Als Grundlage dienen die Fragen aus Abschnitt 5. Die Teilnehmer sollen dazu aufgefordert werden, die Demonstration an ihren Computern nachzustellen. Im Planungssystem stehen zu diesem Zweck 5 Logins ,,spielwiese1“ bis ,,spielwiese5“ zur Verfügung[1].

(4) **Anwendung der Features bei den Testaufgaben:**
Wie führe ich Änderungen an Ressourcen, Reihenfolge und Startzeiten von Jobs durch? + Wie ändere ich Optimierungsziele für best. Jobgruppen?

[1] Der Test wird auf mehrere Termine verteilt, an denen jeweils 5 Testpersonen (aus jeder Testgruppe ein Teilnehmer) eingeladen werden.

(5) Der Betreuer erklärt, wie eine Testaufgabenstellung zu lesen ist und welche Aktionen vor und nach jedem Test auszuführen sind (siehe Abschnitt 3). Er erklärt das Einloggen, Bearbeiten und Speichern von Tests anhand einer bestimmten Testaufgabe.

(6) Der Betreuer erklärt abschließend, dass

   (a) für jeden Test 20 min. zur Verfügung stehen

   (b) während des Tests Fragen gestellt werden können und außerdem das im Menü verlinkte Hilfe-Dokument genutzt werden kann

   (c) möglichst alle Features benutzt werden sollen, die bei der jeweiligen Aufgabe zur Verfügung stehen

   (d) Tests nur unter der Aufsicht der Testperson begonnen und beendet werden

   (e) ein Test erst dann als erfüllt gilt, wenn alle Jobs **konfliktfrei** verplant wurden und sich keine Jobs mehr in der Ablage befinden

   (f) jeder Test mit einem Screen-Capture-Programm aufgezeichnet wird

   (g) jegliches Feedback ausdrücklich erwünscht ist

Die Einweisung sollte 20 min. nicht überschreiten.

2.3 Betreuung der Tests

Protokollformular:

Der Betreuer füllt für jeden Teilnehmer ein Protokollformular aus. Testname und Login werden anhand der Testgruppe vorher ausgefüllt. Das Passwort lautet immer ,,tester“.

Im Abschnitt Bemerkungen sind einzutragen:

- Fragen, die der Teilnehmer nach/vor einem Test gestellt hat
- Probleme, die aufgetreten sind (Bugs oder Systemabsturz, sodass der Test abgebrochen werden musste)
- Überschreitung der maximalen Zeit (20 min.)
- sonstige Hinweise und Feedback des Teilnehmers

Test	Login	Beginn	Ende	Bemerkungen

Testaufgaben:

Dem Teilnehmer sind die Testaufgaben aus Abschnitt 4 und die Hinweise aus Abschnitt 3 vorzulegen. Die Testaufgaben aus Abschnitt 4 sind abhängig von der Testgruppe nach der in Abschnitt 2.1 vorgegebenen Reihenfolge zu sortieren. Außerdem muss für jeden Test das entsprechende Login eingetragen werden.

Der Teilnehmer soll ermutigt werden, die für den jeweiligen Test freigeschalteten Features einzusetzen. Daher wird über jeder Testaufgabe die von der Gruppe abhängige Konfiguration angezeigt:

Feature	Aktiviert
Vollständige automatische Planung	X
Automatische Planung Gruppen	
Automatische Planung Ressourcen	
Fixierung	X
Hervorhebung gültiger Einfügepositionen	
Aufheben von Konflikten („Einordnen“)	X

Wenn ein Teilnehmer mit einem Test fertig ist, prüft der Betreuer, ob der Test gespeichert wurde und ob alle Jobs eingeplant wurden. Vor/nach jedem Test wird das ScreenCapture-Programm gestartet/beendet (F9).

2.4 Einrichtung der PCs

(1) Der Betreuer muss sich mit seinem Login an den Testrechnern anmelden. Dieses Login wird dann auch von der Testperson verwendet.

(2) Die Tests werden mit Mozilla Firefox durchgeführt. Auf jeden Computer muss vorher die portable Version von Firefox installiert werden:
http://www.chip.de/downloads/c1_downloads_auswahl_15821029.html?t=1353662036&v=3600&s=6cf33a8b4ccf92c27efc7391e49bbf43

(3) Die Webseite http://comores.inf.hs-zigr.de/eus_flottenplanung muss auf jedem Rechner geöffnet werden, sodass das Login-Fenster erscheint.

(4) Das Screen-Capture-Programm muss auf die Rechner kopiert werden. Durch Drücken von F9 wird getestet, ob das Programm zur Aufzeichnung bereit ist. Nochmals F9 drücken, um die Testaufzeichnung zu beenden, das Testvideo sollte auf dem Desktop erscheinen.

(5) Auf jedem Arbeitsplatz werden die Testaufgaben, die Teilnehmerhinweise, das Protokoll und ein Kugelschreiber ausgelegt.

2.5 Aufzeichnung der Tests

Die Durchführung der Tests wird mit dem portablen Screen-Capture-Programm „AutoScreen-Recorder 3.0 Free“ aufgezeichnet. Jede Testaufgabe wird einzeln aufgenommen. Die Videogröße soll bei einer maximalen Aufgabendauer von 20 min. 1GB nicht überschreiten. Dies wird mit folgenden Einstellungen erreicht:

- Codec: Microsoft Video 1
- Zeit / Bildqualitätsfaktor: 0.5

3 Hinweise für die Teilnehmer

3.1 Allgemeines

Danke, dass Sie sich die Zeit nehmen, um an unserem Usability-Test teilzunehmen! Ihre Teilnahme liefert uns wertvolle Erkenntnisse für die Gestaltung von Planungssystemen.

Denken Sie bitte daran: Wir testen nicht Sie, sondern Sie testen die Anwendung! Sie können keine Fehler machen. Sie erproben lediglich unsere Oberfläche.

Haben Sie daher keine Scheu, Fragen zu stellen oder Kritik zu äußern.

3.2 Anmeldung am System

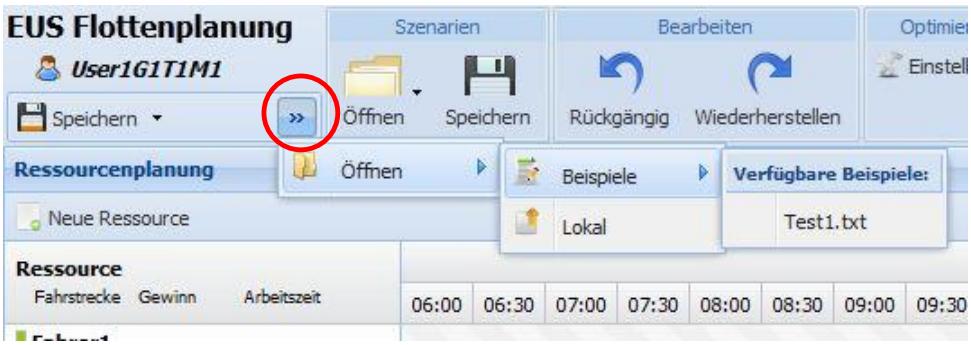
Das System befindet sich unter der URL http://comores.inf.hs-zigr.de/eus_flottenplanung. Für die Tests wird der Browser „Mozilla Firefox“ verwendet.

Vor jedem Test melden Sie sich dort mit dem in der Testaufgabe verzeichneten Login an. Das Passwort lautet: „Tester“.

Wenn Sie mit einem Test fertig sind, rufen Sie einen Betreuer. Sie melden sich dann ab, indem Sie auf den „Abmelden“-Button klicken.

3.3 Öffnen und Speichern der Tests

In der Testaufgabenstellung ist der Testname angegeben. Öffnen Sie den Test im Menü unter **Öffnen → Beispiele:**



Speichern Sie zwischendurch mehrmals den Test:

Wenn Sie mit dem Test fertig sind, **speichern Sie den Test nochmals.**

Bevor Sie sich ausloggen, melden Sie sich bitte bei einem Betreuer.

4 Testaufgaben

Login	Siehe Protokollformular
Test	Test1.txt
Aufgabenstellung	**Änderungen Fahrer4:** **Auftrag 25** soll zwischen **11:00** und **11:15** von **Fahrer4** ausgeführt wer-

	den. **Auftrag 6** soll zwischen **13:00** und **13:30** von **Fahrer4** ausgeführt werden. 1) Ändern Sie die Route den Vorgaben entsprechend. 2) Minimieren Sie die Fahrstrecke der Route. **Änderungen Fahrer2:** **Auftrag 10** soll um **9:30** von **Fahrer2** ausgeführt werden. **Auftrag 16** soll zwischen **14:30** und **16:00** von **Fahrer2** ausgeführt werden. 1) Ändern Sie die Route den Vorgaben entsprechend. 2) Minimieren Sie die Fahrstrecke der Route. **Änderungen Fahrer3:** **Auftrag 24** von **Fahrer3** muss um **10:15** stattfinden.

Login	Siehe Protokollformular
Test	Test2.txt
Aufgabenstellung	**Auftrag 2** soll von **Fahrer4** zu **Fahrer1** verlegt werden. 1) Verschieben Sie **Auftrag 2** zu **Fahrer1.** 2) Minimieren Sie die Fahrstrecke für **Fahrer1.** 3) Minimieren Sie die Fahrstrecke für **Fahrer4.** **Auftrag17** soll von **Fahrer2** zu **Fahrer3** verlegt werden. 1) Verschieben Sie **Auftrag17** zu **Fahrer3** und stellen Sie eine gültige Tour her. Um eine gültige Lösung zu erhalten, müssen ggf. einige Aufträge auf andere Fahrer verteilt werden. **Die Aufträge 12, 14, 15 und 16 und 18 müssen bei Fahrer3 bleiben. Achten Sie darauf, dass diese Aufträge nicht durch eine Optimierung verschoben werden!** 2) Minimieren Sie die Fahrstrecken der geänderten Touren.

Login	Siehe Protokollformular
Test	Test3.txt
Aufgabenstellung	Planen Sie alle Aufträge in der Ablage so ein, dass die Gesamtfahrstrecke minimal ist. Die Reihenfolge der bereits eingeplanten Aufträge darf nicht verändert werden. Andere Aufträge dürfen aber dazwischen geschoben werden.

Login	Siehe Protokollformular
Test	Test4.txt
Aufgabenstellung	**Auftrag 5** liegt in der Ablage. Planen Sie **Auftrag 5** in die Tour von **Fahrer2**. Um eine gültige Tour herzustellen, müssen ein oder mehr andere Aufträge von **Fahrer2** auf andere Fahrer verlegt werden. Planen Sie die restlichen Aufträge in der Ablage ein. Ändern Sie nichts mehr an den Touren von **Fahrer1** und **Fahrer2**. Versuchen Sie immer, eine möglichst geringe Gesamtfahrstrecke zu erzeugen.

Login	Siehe Protokollformular
Test	Test5.txt
Aufgabenstellung	Verteilen Sie alle Aufträge in der Ablage auf die 4 Fahrer. Sie dürfen die bereits eingeplanten Aufträge umplanen. Es gelten folgende Einschränkungen: 1) **Auftrag7** und **Auftrag10** müssen von **Fahrer1** übernommen werden. 2) **Auftrag22** muss von **Fahrer4** übernommen werden. Versuchen Sie, durch geschickte Einplanung und Umplanung eine minimale Fahrstrecke zu erzeugen!

Login	Siehe Protokollformular
Test	Test6.txt
Aufgabenstellung	Planen Sie alle Aufgaben so ein, dass jeder Fahrer ca. **13 Uhr wieder im Depot ist.** Beachten Sie folgende Einschränkungen: 1) Die Aufträge „**Auftrag18“ und „Kopie Auftrag18“ sollen gleichzeitig von Fahrer1 und Fahrer4 um 9:30 ausgeführt werden.** 2) Richten Sie die Touren so ein, dass **Auftrag 15** zeitlich vor **Auftrag 10** liegt.

5 Themen für die Einweisung

- Wie kann ich Aufträge manuell planen?
- Was bedeuten die Hintergrundfarben beim Drag and Drop?
- Kann ich einen einzelnen Auftrag automatisch einplanen lassen?
- Kann ich mehrere Aufträge auf einmal einplanen lassen?
- Wie kann ich alle Aufträge automatisch einplanen lassen?
- Kann ich einen automatisch erzeugten Plan verändern?
- Kann ich die Routen der Fahrer auch einzeln optimieren?
- Ich möchte gern alles neu optimieren, aber einige Aufträge dürfen nicht verschoben werden.
- Ich habe die Fahrzeit automatisch optimiert, aber die Fahrer sind ungleichmäßig ausgelastet.
- Im Plan wird bei manchen Jobs eine Fehlermeldung angezeigt. Was muss ich tun? / Wie kann ich Fehler bei der Planung beseitigen?
- Wie kann ich verhindern, dass ein Fahrer bestimmte Aufträge zugeteilt bekommt?
- Kann ich eine Tour auch im Ganzen einem anderen Fahrer zuordnen?
- Warum werden in der Karte nur die Touren für einen Tag angezeigt?

Anhang B

Testergebnisse

Die folgenden Abbildungen zeigen Ergebnisse des Anwendertests für die Fallstudie der Flottenplanung. Es wird die im ersten und zweiten Testdurchgang mit unterschiedlichen Systemkonfigurationen erzielte Qualität der Pläne (Gesamtfahrzeit) dargestellt.

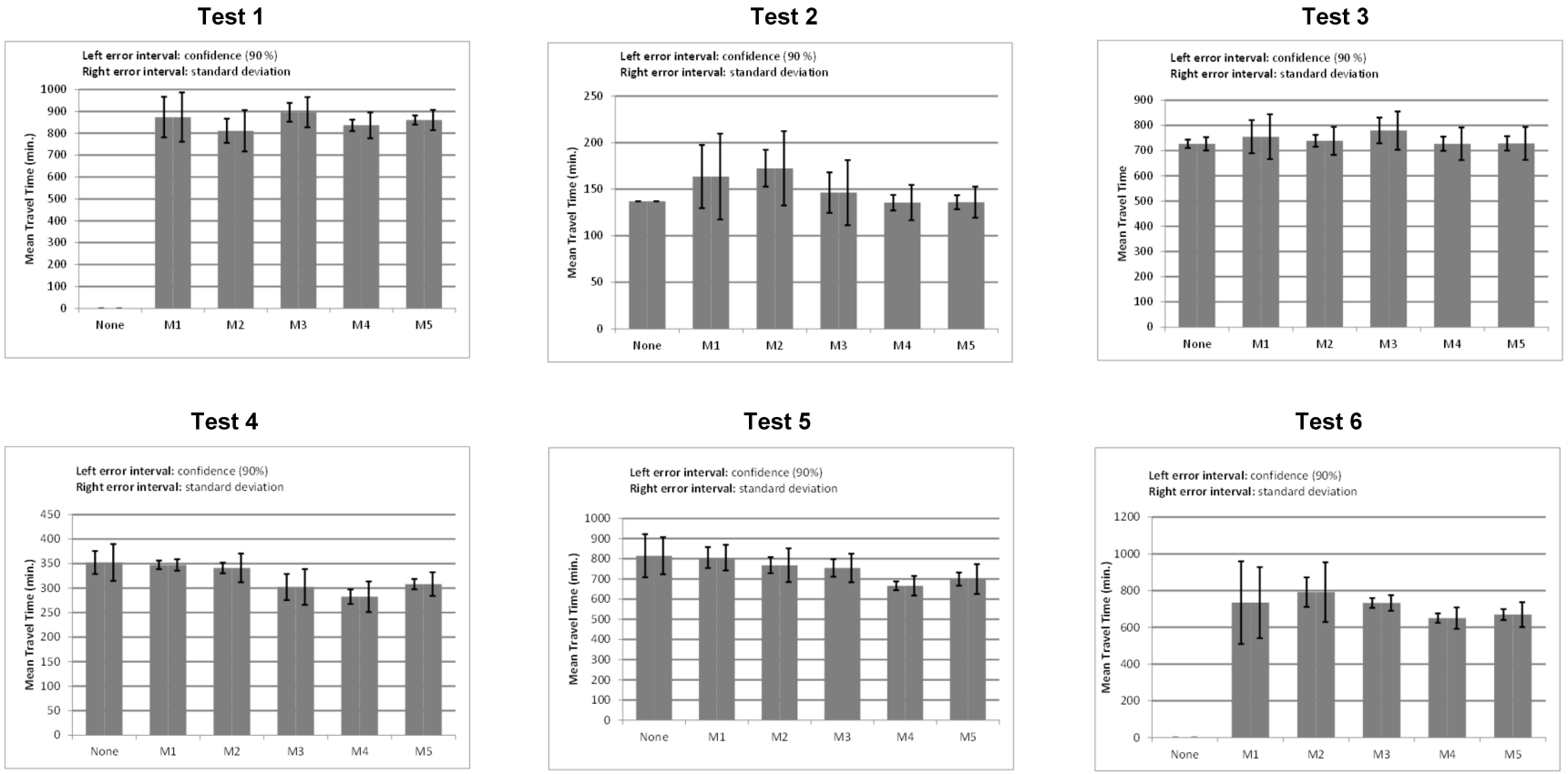


Abbildung B.1: Übersicht über die durchschnittlichen Gesamtfahrzeiten, die pro Teilnehmer mit den einzelnen Systemkonfigurationen für Test 1 bis Test 6 erzielt wurden (Testdurchgang 1)

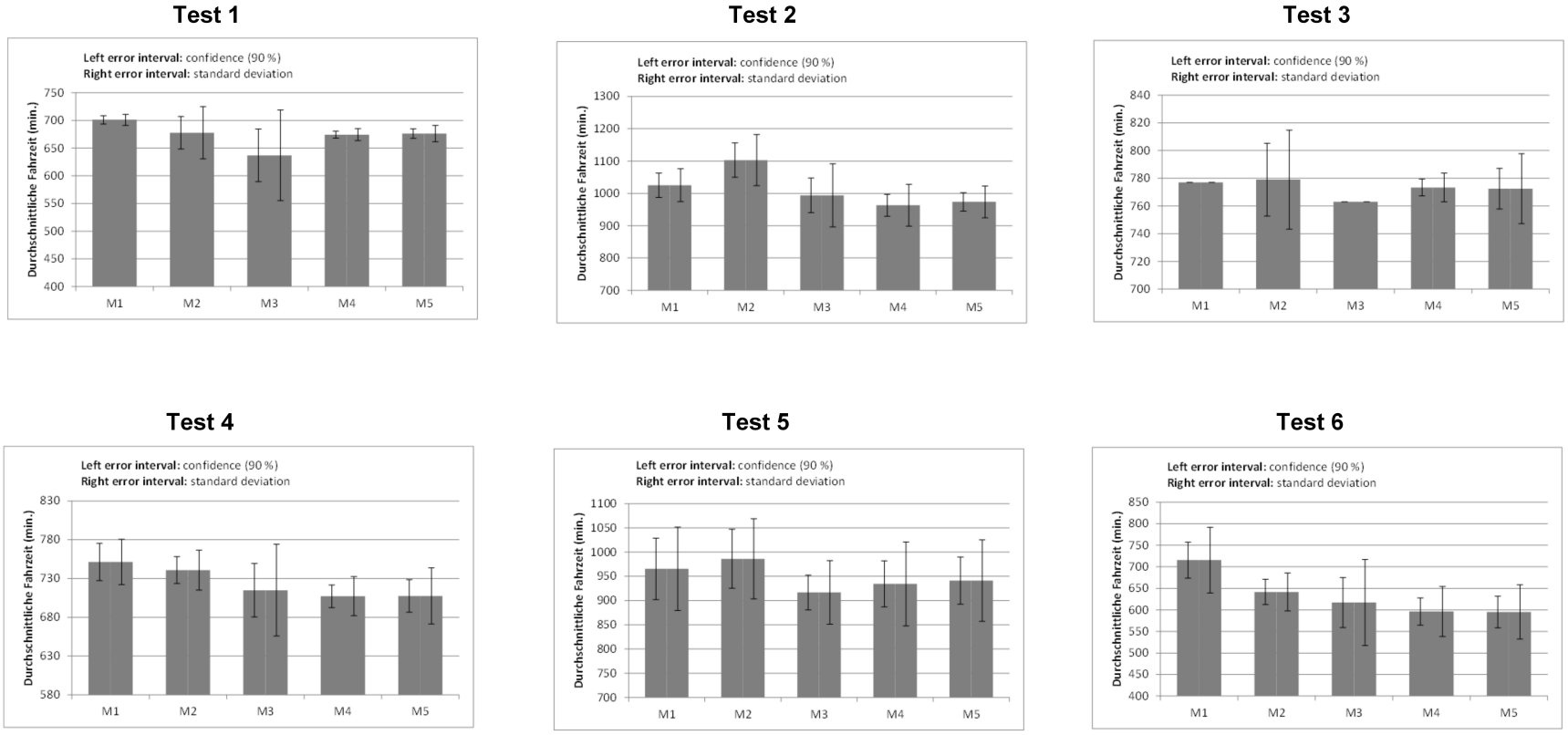


Abbildung B.2: Übersicht über die durchschnittlichen Gesamtfahrzeiten, die pro Teilnehmer mit den einzelnen Systemkonfigurationen für Test 1 bis Test 6 erzielt wurden (Testdurchgang 2)

Anhang C

Bezeichnungen aus der allgemeinen Modellstruktur

Die folgenden Bezeichnungen sind Bestandteil der in Abschnitt 2.2 beschriebenen allgemeinen Modellstruktur eines Planungsproblems:

n	die Anzahl der Aufgaben
u	die Anzahl der Ressourcen
AS	die Menge aller Aufgaben
RS	die Menge aller Ressourcen
P	der Planungshorizont
a	eine einzelne Aufgabe aus AS
a_{start}, a_{end}	Zeitpunkt für den Beginn bzw. das Ende der Ausführung einer Aufgabe
res	eine einzelne Ressource aus RS

Allgemeine Bezeichnungen für die Beschreibung eines CSPs (Abschnitt 2.1):

V	die Menge der Variablen
D	die Menge der Domänen der Variablen
C	die Menge der Randbedingungen (Constraints)
S_i	die Menge von Variablen, über die eine Randbedingung $c_i \in C$ definiert ist